UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Sistema de recomendación de campeones del League of Legends: estudio de representaciones de datos y nuevos métodos de evaluación de árboles de Monte Carlo

Autor: Iris Álvarez Nieto
Tutor: Lara Quijano Sánchez

junio 2021

Iris Álvarez Nieto
Sistema de recomendación de campeones del League of Legends: estudio de representaciones de datos y nuevos métodos de evaluación de árboles de Monte Carlo

Iris Álvarez Nieto

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

Me gustaría agradecer a mi tutora Lara por acompañarme en este proceso. A mi familia y a mis amigos por su apoyo. A Tina, Nacho y Adrián. Y en especial, mis más sinceros agradecimientos a Carmen y a mi madre, por hacer siempre todo más fácil.

Resumen

Los videojuegos multijugador de arena de batalla en línea o MOBA (*Multiplayer Online Battle Arena*) son uno de los tipos de videojuego más jugado actualmente. Los usuarios mensuales de uno sólo de ellos (*League of Legends*) alcanzan los 115 millones. En una partida de MOBA, dos equipos de 5 jugadores se enfrentan para destruir la base del equipo contrario. Antes de empezar cada equipo elige, entre más de 100 posibilidades, 5 personajes, llamados también campeones. A este proceso se le conoce como *draft* y es esencial para ganar. Una composición fuerte debe contar con personajes que se complementan entre ellos y contrarrestan a los enemigos.

Anteriormente, se han desarrollado diversos trabajos de investigación ya sea para predecir la probabilidad de victoria de una composición final como para recomendar uno a uno los campeones a elegir. Sin embargo, existe una carencia común, ninguno emplea más datos a parte de los ids de los campeones para representarlos. Esto plantea la posibilidad de investigar diferentes formas de representación de los datos.

El presente trabajo emplea el mejor modelo del estado del arte: el recomendador de campeones mediante árboles de búsqueda de Monte Carlo y redes neuronales. Se plantean dos nuevas representaciones que consideran las estadísticas particulares de cada campeón y se comparan con la actual representación de datos que sólo considera los ids. Como caso de estudio, se emplean los datos del videojuego *League of Legends* (LoL) al ser el MOBA más popular actualmente.

Los valores obtenidos con redes neuronales son parecidos usando la representación clásica de ids y una de las nuevas configuraciones. Respecto a los árboles, ninguno es significativamente mejor que el resto. Por último, se evalúa si las composiciones son realistas. Los mejores resultados se obtienen con las nuevas representaciones y podemos afirmar que son casi realistas, con al menos 4 campeones asignados correctamente.

Además en este trabajo se han abordado otras limitaciones de los trabajos relacionados, como son: la evaluación de árboles de búsqueda cuando no se puede determinar el ganador al observar el resultado final y carencias en trabajos del LoL respecto a la base de datos utilizada y a la metodología.

Palabras clave

MOBA, League of Legends, draft, árbol de búsqueda Monte Carlo, redes neuronales

Abstract

Multiplayer online battle arena or MOBA are one of the most widely played types of video games today. The monthly users reported by just one of them (*League of Legends*) reach 115 million. In a MOBA game, two teams of 5 players face off to destroy the opposing team's base. Before starting, each team chooses, among more than 100 possibilities, 5 characters, also called champions. This process is known as *draft* and is essential to winning. A strong composition features characters that complement each other and counter enemies.

Previously, various research works have been developed either to predict the probability of victory of a final composition or to recommend one by one the champions to choose from. However, there is a common lack, none uses more data than the champions ids to represent them. This raises the possibility of investigating different forms of data representation.

The present work uses the best model in the state of the art: the champion recommender using Monte Carlo tree search and neural networks. Two new representations are proposed that consider the particular statistics of each champion and are compared with the current representation of data that only considers the ids. As a case study, data from the video game *League of Legends* (LoL) is used as it is currently the most popular MOBA.

The values obtained with neural networks are similar using the classical representation of ids and one of the new configurations. Regarding trees, none is significantly better than the rest. Finally, it is evaluated if the compositions are realistic. The best results are obtained with the new representations and we can affirm that they are almost realistic, with at least 4 champions correctly assigned.

In addition, this work has addressed other limitations of the related works, such as: the evaluation of search trees when the winner cannot be determined by observing the final result and deficiencies in LoL works regarding the database used and the methodology.

Keywords

MOBA, League of Legends, draft, Monte Carlo tree search, neural networks

Índice

Listas

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

Los videojuegos multijugador de arena de batalla en línea o MOBA (*Multiplayer Online Battle Arena*) son uno de los tipos de videojuego más jugado actualmente y es el género más popular de los *eSports*. [1]

Entre los MOBAs, el más popular es *League of Legends* (LoL). Aunque han pasado 12 años desde su lanzamiento, cuenta con 115 millones de usuarios activos al mes y es uno de los juegos más populares en la plataforma de transmisiones en directo Twitch. [2]

En un MOBA, dos equipos de 5 jugadores luchan por destruir la base del contrincante. Consta de dos fases: la primera es puramente estratégica, donde cada jugador elige al personaje, o campeón, que controlará (*draft*). En la segunda, se desarrolla el juego, en el que se intenta destruir la base del contrincante colaborando con tu equipo, consiguiendo objetivos y protegiendo tus estructuras.

Entre los factores determinantes para ganar o perder una partida están la habilidad de cada jugador, lo fuertes que sean los campeones individualmente, la relación de los jugadores del equipo y la composición obtenida tras la primera fase de la partida. Esta última es tan esencial que muchos jugadores del LoL abandonan partidas antes de jugar la segunda fase porque saben que es muy complicado ganar [3]. También, en la competición profesional los *drafts* se comentan y analizan tanto para predecir el resultado de la partida como para explicarlo. En algunos casos, la diferencia de calidad de las composiciones es tan grande que es muy poco probable ganar la partida [4].

En los MOBAs suele haber más de 100 personajes entre los que elegir. En el caso del LoL encontramos 154 campeones [5], por lo que existen aproximadamente $3{,}87 \times 10^{17}$ ($C_{154}^{5} \times C_{149}^{5}$, escogemos 5 campeones entre los 154 y 5 entre los restantes) posibles composiciones.

[1] https://www.marca.com/esports/2019/04/02/5ca2c4d1e5fdeab1618b45d5.html

[2] https://screenrant.com/league-legends-twitch-popular-moba-streamers-lol-championship/

[3] https://www.esportmaniacos.com/guias/cuando-dodgear-en-soloq-en-league-of-legends/

[4] https://www.invenglobal.com/articles/12530/worlds-2020-reapered-breaks-down-gens-drafts-in-0-3-loss-to-g2-they-were-an-old-stubborn-hermit

[5] La versión empleada en este trabajo es la 11.7, que es la última que cuenta con 154 campeones antes de que se añada a Gwen.

Debido a su importancia y complejidad, en este trabajo se realiza un estudio exhaustivo de técnicas para la recomendación de campeones en la fase del *draft*, estudiando los aspectos comunes en anteriores estudios, y las técnicas y metodologías que a día de hoy han resultado más efectivas. A partir de ese estudio, se plantean distintas alternativas novedosas, especialmente en la codificación de los jugadores y se estudia su impacto en los resultados.

1.2. Objetivos

Como se ha introducido, este trabajo estudia el efecto de emplear otras configuraciones de datos en el desempeño de modelos utilizados en el *draft* dentro del MOBA más popular, *League of Legends*. Se pretende resolver una carencia común de todos los trabajos anteriores, que sólo emplean los ids de los campeones para representarlos.

En la selección de campeones o *draft*, cada equipo veta y escoge 5 campeones tratando de conseguir formar una composición que pueda ganar al rival. Los trabajos desarrollados anteriormente sobre este tema se pueden dividir en dos grupos dependiendo de si se basan en la frecuencia de selección [1–4] o la probabilidad de victoria [5–17]. En la primera categoría, los modelos son entrenados para sugerir campeones que usualmente se eligen. En la segunda categoría, se desarrollan trabajos que predicen el equipo victorioso a partir de las composiciones de los equipos o que recomiendan campeones que aumentan la probabilidad de ganar. Si se comparan los datos que usa cada postura, mientras la primera sólo tiene en cuenta la composición de cada equipo, la segunda también quién fue el ganador.

Generalmente los trabajos emplean los datos del videojuego Dota 2 y sólo en ocasiones los de otro videojuego como el LoL o *Honor of Kings*. Respecto a la codificación, es muy limitada. Es generalizado emplear únicamente los ids de los campeones y los pocos trabajos que se salen de esta norma añaden atributos que aportan un matiz a este enfoque, más que plantear uno nuevo.

Basándose en los trabajos de la segunda categoría, este trabajo usa el mismo enfoque para realizar recomendaciones de campeones, maximizando la probabilidad de victoria dados los elegidos hasta entonces.

Se emplean los métodos con mejores resultados hasta el momento, que se detallan en la sección 2.2. En concreto, se tratan de árboles de búsqueda Monte Carlo (*Monte Carlo Tree Search* (MCTS)) y redes neuronales [5, 6].

El videojuego utilizado como caso de estudio es el LoL, al ser el MOBA más popular y, por tanto, donde las aportaciones serán más valiosas. Este videojuego aporta mayor complejidad que los otros empleados, al contar con mayor número de campeones. El Dota 2 cuenta actualmente con 121 campeones, sin embargo, los datasets empleados anteriormente alcanzan los 114 campeones como

máximo. En el Honor of Kings, el número de campeones ronda los 100. Mientras, en este trabajo se consideran 154 campeones del LoL. Además, los trabajos anteriores del LoL no tienen bases de datos públicas, por lo que desarrollamos una propia.

Respecto a la codificación, como se ha mencionado anteriormente, todos los trabajos anteriores se centran únicamente en una forma de representar los datos. Aquí se plantean dos nuevas codificaciones que emplean las estadísticas particulares de los campeones para representarlos, en vez de hacerlo únicamente con sus ids. La primera codificación representa las estadísticas acumuladas de cada equipo. La segunda, emplea las estadísticas individuales de cada campeón dependiendo de su equipo y su posición.

Además, los trabajos que recomiendan campeones no estudian si las composiciones obtenidas son realistas o no. En este trabajo, se explica cómo son las composiciones usuales y qué requisitos mínimos tiene que cumplir para ser realista en la sección 2.1 y se plantea una solución en 4.3.

Por último, cabe señalar que, aunque los mejores resultados hayan sido obtenidos usando Monte Carlo junto con redes neuronales, la evaluación del modelo no ha sido totalmente correcta. Para evaluar un árbol Monte Carlo se observa el estado final resultante de realizar las acciones sugeridas. Sin embargo, en este ámbito no se puede conocer al ganador al observar el *draft*. No es lo mismo evaluar el resultado final de una partida de tres en raya, donde el ganador es observable, que evaluar una composición, donde sólo podemos hablar de probabilidades de ganar. En este caso, hasta que no se juegue la partida no sabremos qué composición es la ganadora.

Los trabajos anteriores resuelven este problema evaluando las composiciones finales con el predictor de victoria que han empleado en el Monte Carlo. El equipo que obtenga la mayor probabilidad de victoria es el ganador. Sin embargo, este enfoque incluye un prejuicio que evaluará favorablemente a las composiciones generadas por el Monte Carlo. Esta problemática se discute y se plantea una solución más imparcial en la sección 4.3.

En resumen, este trabajo presenta las siguientes contribuciones:

- Primer dataset público del *draft* del LoL. Los trabajos sobre composiciones basados en el LoL no publican las partidas empleadas y además su muestra es reducida [2,7,15]. El dataset obtenido es más extenso que los que se pueden encontrar en internet cuando las partidas tienen que ser de la misma versión del juego y contar con roles correctos.
- Uso de configuraciones de datos nunca utilizadas. Con ellas, se optimizan los hiperparámetros y entrenan tres redes neuronales distintas en total.
- Estudio de la eficacia del Monte Carlo dependiendo de la red neuronal usada.
- Se resuelve una carencia de otros trabajos al aplicar Monte Carlo [5,6], donde la evaluación de los estados finales no está libre de prejuicio.
- Se estudia si las composiciones generadas son realistas.

1.3. Organización de la memoria

La memoria está estructurada de la siguiente forma. Primero, se explica el videojuego *League of Legends*, centrándose en la selección de campeones y se presenta un resumen general de los trabajos realizados anteriormente sobre MOBAs. Se señalan las aportaciones y limitaciones de cada uno de ellos y aquellos que se tomarán como referencia. Después, se plantea la formulación del problema y se explican las configuraciones de datos y modelos empleados. Tras ello, se expone la experimentación, incluyendo la toma de datos y los detalles del proceso seguido. Por último, se estudian los resultados y aportaciones conseguidas, y se plantean las posibles futuras mejoras.

2

Estado del arte

En esta sección, se explican las bases del videojuego de estudio *League of Legends* centrándose en específico en la materia relacionada con los campeones. Se expone cómo se eligen, qué les diferencia y qué es importante a tener en cuenta al crear una composición. Así mismo, se comentan los trabajos relacionados con el *draft* de MOBAs.

2.1. League of Legends

League of Legends es un videojuego multijugador de arena de batalla en línea donde dos equipos de cinco jugadores cada uno se enfrentan para ver quién destruye primero la base del rival. Cada jugador controla un campeón para, junto a sus compañeros, destruir objetivos y obtener ventajas que les lleven a la victoria.

El juego se divide en dos fases. En la primera, se desarrolla el *draft*. Primero cada equipo veta 5 campeones que no podrán ser jugados en esa partida. Después, se elige por turnos la composición de cada equipo. Se alternan siguiendo la secuencia 1-2-2-2-2-1, es decir, el primer equipo elige un campeón, después el rival elige dos, y así hasta formar los dos equipos de 5 personajes. En la segunda fase, los equipos aparecen en el mapa y empieza la partida para ver quién es el primero en destruir la base enemiga. Los detalles de la partida se explican en el Apéndice A.

En el LoL existen 154 campeones entre los que elegir. Cada campeón cuenta con unas estadísticas y habilidades particulares que le hacen único.

Entre las estadísticas, la más esencial es la cantidad de vida que poseen para resistir ataques. Los ataques pueden ser de dos tipos de daño: físico y mágico. Cada campeón tiene una cantidad de daño que inflige de cada tipo, así como de resistencia a este (armadura y resistencia mágica, respectivamente). También, cuentan con velocidad de movimiento y velocidad y rango del ataque básico. Todas las estadísticas anteriores junto con otras, tienen un valor base al comienzo de la partida y van aumentando según suba de nivel el campeón. Al principio, los campeones son de nivel 1 y pueden llegar al nivel 18 como máximo. Por cada nivel, se suma una cantidad constante a cada estadística base.

Los campeones pueden atacar mediante ataques básicos o habilidades. Cada campeón cuenta con 4 habilidades y una pasiva. El ataque básico se lanza haciendo clic en un campeón, las habilidades se lanzan pulsando las teclas Q, W, E y R, y la pasiva, como su nombre indica, afecta al campeón sin que el jugador haga nada. Las habilidades no pueden lanzarse todo el rato pues tienen un tiempo de espera para poder volver a lanzarlas. Además, muchos campeones no pueden lanzar las habilidades de forma gratuita, consumen un recurso específico que tienen que gestionar. El recurso más común es el maná, del cual cada campeón cuenta con una cantidad específica, como lo hace con la vida. Mientras la vida se consume cuando reciben daño, el maná se gasta cuando lanzas una habilidad.

Dependiendo de las estadísticas y habilidades anteriores, los campeones se pueden clasificar en las siguientes categorías [1] [2]:

- Asesinos. Campeones con alta movilidad y daño explosivo contra un solo campeón. No tienen demasiada vida ni resistencia al daño y dependen de ejecutar bien un combo de habilidades. Son muy buenos infiltrándose en territorio enemigo, matar a un campeón prioritario y volver a un lugar seguro. Son campeones rápidos a los que les beneficia atacar por sorpresa a enemigos aislados.
- Luchadores. Campeones de ataque cuerpo a cuerpo que destacan por su capacidad tanto de resistir como infligir daño. Destacan en combates prolongados donde pueden hacer uso de ambas cualidades resistiendo e infligiendo daño prolongado.
- Magos. Campeones de daño mágico que cuentan con habilidades de gran rango, que afectan en área o controlan al adversario (por ejemplo impidiendo el movimiento o el lanzamiento de habilidades). Usan el rango de sus habilidades para matar al enemigo sin exponerse a riesgos pero dependen de ellas para infligir suficiente daño. Pueden ocupar un rol más de utilidad mediante sus habilidades de control o ser una de las principales fuentes de daño en peleas. Su mayor debilidad es su falta de movilidad y de resistencia al daño.
- Tiradores. Campeones de daño a distancia mediante ataques básicos. Son la principal fuente de daño continuado. Son buenos en peleas largas, destruyendo objetivos y necesarios para acabar con los campeones con más vida. Igual que los magos, tienen poca vida y resistencia, pero a diferencia de ellos, cuentan con menor movilidad y menos habilidades que permitan controlar al rival. Por tanto, suelen necesitar a alguien protegiéndolos que les permita sobrevivir.
- Apoyos. Campeones que destacan por la utilidad que otorgan al resto del equipo mediante protección, curación o control del adversario. Suelen acompañar a los tiradores pues se complementan entre ellos: el apoyo, no tiene mucho daño, pero protege al tirador, que es muy débil solo e inflige gran cantidad de daño.
- Tanques. Campeones que tienen mucha vida y resistencia y cuentan con habilidades que

[1] https://euw.leagueoflegends.com/es-es/champions/

[2] https://leagueoflegends.fandom.com/wiki/Champion_classes

controlan al adversario. No tienen mucho daño pero permiten al resto de su equipo realizarlo. Su objetivo es resistir y afectar al equipo rival en primera línea mientras su equipo, detrás de él, proporciona el daño necesario.

Un campeón puede entrar en una o más categorías de las anteriores. Un ejemplo de campeón es Lux. Es un mago pues su daño es sobre todo mágico y tiene habilidades de largo alcance que afectan en área (E y R) y controlan al adversario (Q y E). También, es un campeón apoyo porque cuenta con una habilidad defensiva (W) y dos habilidades de control del adversario (Q y E). En la imagen 2.1 se pueden observar las habilidades de Lux y su cantidad de vida y maná.

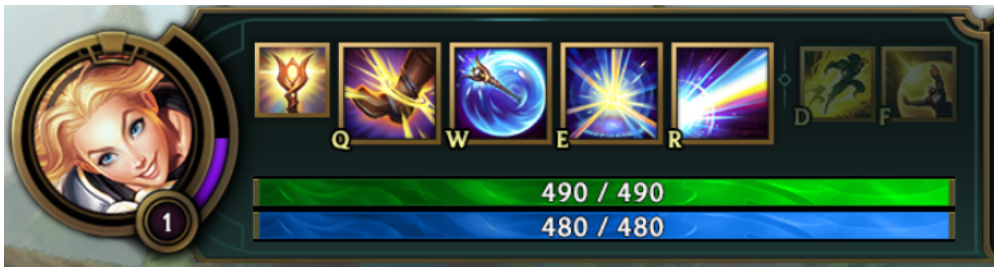

Figura 2.1: Información del campeón Lux en una partida. De izquierda a derecha vemos: su pasiva y sus habilidades (representadas por cuatro cuadrados). Debajo de ellas se sitúan dos barras: la primera es la cantidad de vida y la segunda, la del maná. *Imagen extraída de Página oficial del League of Legends.*

Las fortalezas y debilidades de las categorías de campeones se deben tener en cuenta para crear combinaciones de ellos que aumenten las primeras y reduzcan las segundas. Una composición clásica cuenta con un tanque para que resista daño, un campeón que aporte daño mágico, un tirador para que inflija daño básico continuado y otros dos campeones que aporten control del adversario, apoyo o sean buenos iniciando peleas.

Además, no se puede escoger los 5 campeones sólo teniendo en cuenta sus categorías. Las composiciones cuentan con 5 posiciones o roles diferentes. Dependiendo de la posición que juegues te sitúas en un sitio concreto del mapa y tu aportación al equipo es distinta. Las posiciones en orden descendente según su posición en el mapa son: calle superior, jungla, calle central, calle inferior y apoyo. La distribución de los roles se puede observar en la imagen 2.2.

Por tanto, además de crear una buena combinación de campeones según sus categorías, cada campeón particular tiene que ser bueno en el rol asignado.

En la calle inferior suele ir un campeón tirador y uno de apoyo. De esta forma, los campeones se complementan y el tirador estará protegido. Se sitúan en la inferior y no en la superior, pues el tirador es bueno obteniendo objetivos y en la inferior se encuentra un objetivo principal (dragones). En la calle central suelen ir magos o asesinos, pues son campeones más capaces de ayudar a otras líneas. Se sitúan en el centro para que puedan ayudar a cualquier línea. En la jungla se juegan campeones que puedan soportar daño e infligirlo así como poder comenzar peleas cuando vayan a ayudar a las calles.

Figura 2.2: Mapa donde se señalan de arriba a abajo las posiciones: calle superior, jungla, calle central, calle inferior y apoyo. El jungla se mueve por todas las zonas entre líneas. En la calle inferior se sitúan el rol con este nombre y el apoyo. *Imagen extraída de Página oficial del League of Legends.*

Por estos motivos, muchos luchadores son buenos junglas. Por último, falta que el equipo cuente con un campeón que soporte mucho daño continuado. Por esta razón, en la parte superior del mapa suelen ir luchadores y tanques.

Por último, además de tener en cuenta lo explicado, hay que considerar qué campeones en concreto se complementan y combinan bien, cuáles mejoran los puntos fuertes de otros o disminuyen sus debilidades y cuáles son especialmente fuertes contra los campeones rivales.

Los recomendadores de campeones [5–12] se desarrollan para tener en cuenta todas estas variables y mejorar los resultados del *draft*. Así, se obtiene ventaja frente al rival desde el comienzo de la partida. Sin embargo, algunos recomendadores no tienen en cuenta la posición de los campeones y no generan composiciones realistas. Esta limitación se explica y trata en detalle en la sección 4.3.

2.2. Trabajos relacionados

Para la realización de este trabajo se ha realizado un estudio bibliográfico. El procedimiento se detalla en el Apéndice B donde se incluyen las consultas, motores de búsqueda, resultados y criterios de selección. Tras este proceso, encontramos 35 trabajos relativos a videojuegos online con estrategia de los cuales 29 son relativos a los MOBAs. Sólo 17 de ellos tratan la composición de campeones,

de los cuales 3 emplean el LoL. El resto de trabajos usan datos de las partidas durante el desarrollo en el mapa para predecir la victoria [18, 19] o para realizar un análisis [20–24], estudian el sistema de emparejamiento [25, 26], identifican jugadores tóxicos [27], realizan una revisión de los trabajos sobre MOBAs [28] o crean composiciones siguiendo una estrategia concreta [29]. En la tabla 2.1 encontramos un resumen de todos los trabajos encontrados.

Los 17 trabajos relativos al *draft* en MOBAs se pueden dividir en dos grupos dependiendo de si se basan en la frecuencia de selección o la probabilidad de victoria.

2.2.1. Frecuencia de selección

En la primera categoría, se emplea como datos la frecuencia con la que se selecciona cada campeón, es decir, se utilizan los datos de las composiciones pero no cuál de ellas ganó [1–4]. Se pretende predecir el siguiente veto o selección a partir del estado actual del *draft*.

Summerville et al. [1] resumen este enfoque como "lo que es probable que se elija, no lo que es necesariamente mejor". Prueban a usar redes de Bayes y redes neuronales LSTM con las secuencias del *draft* del Dota 2. Más tarde, Yu et Al. [3] mejoran el modelo LSTM anterior usando Bi-LSTM. También, Gourdeau y Archambault [4] mejoran los resultados de Summerville et al. usando redes neuronales generativas antagónicas y discriminativas. Por último, Hong et al. [2] realizan el mismo enfoque y comparan los resultados de usar redes neuronales o árboles aleatorios pero en el LoL.

De estos trabajos se pueden sacar ideas de modelos y codificaciones que se pueden utilizar. Sin embargo, no tienen en cuenta si las composiciones tienen mayor probabilidad de ganar. Como este trabajo pretende recomendar campeones que aumentan esta probabilidad, no pertenece a esta categoría.

2.2.2. Probabilidad de victoria

Al contrario que la visión anterior, que sólo tiene en cuenta las composiciones, en esta segunda categoría también se considera cuál de ellas resulta victoriosa. Se encuentran predictores de la composición ganadora y recomendadores que forman campeón a campeón la composición con mayor probabilidad de victoria. Este trabajo entra en esta última división, al tratarse de un recomendador.

Los predictores de victoria emplean el conjunto de las composiciones y el resultado del partido para entrenar modelos que predigan qué equipo será el victorioso. En esta división se encuentran los siguientes trabajos.

Agarwala y Pearce [17] utilizan regresión lineal para entrenar tres modelos donde prueban a variar los datos de entrada. Se tienen en cuenta los ids de los campeones y las estadísticas conseguidas por cada uno de ellos al acabar la partida (como el número de muertes). El modelo que emplea úni-

Tabla 2.1: Trabajos encontrados en el estudio bibliográfico.

Documento	Tema	Dominio	Juego	Métodos	Datos	Tamaño base de datos
Chen et al. [5]	Recomendador	MOBA	Honor of Kings	MCTS PUCT y redes neuronales	Ids de campeón y victoria	60 millones de partidas
Chen et al. [6]	Recomendador	MOBA	Dota 2	MCTS UCT y redes neuronales	Ids de campeón y victoria	3.056.596 partidas
Da Costa Oliveira et al. [7]	Recomendador	MOBA	LoL	Minimax	Ids de campeón, posiciones y victoria	Partidas de la competición CBLoL 2017
Hanke y Chaimowicz [8]	Recomendador	MOBA	Dota 2	Reglas de asociación y redes neuronales	Ids de campeón y victoria	-
OpenAI [9]	Inteligencia artificial	MOBA	Dota 2	Aprendizaje profundo por refuerzo y minimax	Observaciones del estado de la partida	-
Ye et al. [10]	Inteligencia artificial	MOBA	Honor of Kings	Aprendizaje profundo por refuerzo, MCTS UCT y redes neuronales	Observaciones del estado de la partida, ids de campeón y victoria	13 millones de partidas para el draft
Conley y Perry [11]	Recomendador	MOBA	Dota 2	Regresión logísticas y KNN	Ids de campeón y victoria	56.691 partidas
Kalyanaraman [12]	Recomendador	MOBA	Dota 2	Algoritmo genético y regresión logística	Ids de campeón y victoria	30.426 partidas
Semenov et al. [13]	Predictor de victoria	MOBA	Dota 2	Regresión logística, Bayes, árboles de decisión potenciados y máquinas de factorización	Ids de campeón, aportación de cada campeón y victoria	5.071.858 partidas
Wang [14]	Predictor de victoria	MOBA	Dota 2	Regresión logística y redes neuronales	Ids de campeón, duración partida y victoria	5.071.858 partidas
Ani et al. [15]	Predictor de victoria	MOBA	LoL	Árboles aleatorios y potenciación del gradiente	Datos de partidas, ids de campeón, vetos, hechizos y victoria	1.500 partidas
Kinkade y Lim [16]	Predictor de victoria	MOBA	Dota 2	Regresión logística y árboles aleatorios	Ids de campeón, sinergias, oponentes y victoria	62.000 partidas
Agarwala y Pearce [17]	Predictor de victoria	MOBA	Dota 2	Regresión logística y PCA	Datos de partidas, ids de campeón y victoria	40.000 partidas

Tabla 2.1: Trabajos encontrados en el estudio bibliográfico.

Documento	Tema	Dominio	Juego	Métodos	Datos	Tamaño base de datos
Summerville et al. [1]	Predictor de selección	MOBA	Dota 2	Redes de Bayes y LSTM	Secuencia del draft	1.518 partidas
Hong et al. [2]	Predictor de selección	MOBA	LoL	Redes neuronales y árboles aleatorios	Secuencia del draft	10.314 partidas
Yu et al. [3]	Predictor de selección	MOBA	Dota 2	Bi-LSTM	Secuencia del draft	4.000 partidas
Gourdeau y Archambault [4]	Predictor de selección	MOBA	Heroes of the Storm y Dota 2	Redes neuronales generativas antagónicas y discriminativas	Secuencias de draft	7.630 partidas del HotS y 10.250 del Dota 2
Pobiedina et al. [20]	Análisis estadístico de factores que afectan a la victoria	MOBA	Dota 2	Tests sobre hipótesis nulas.	Datos de partidas y jugadores	87.204 partidas y 138.101 jugadores
Ward et al. [30]	Construcción de mazos de cartas	Juego de cartas de estrategia	Magic: the Gathering	Heurísticas, Naive Bayes y redes neuronales	Secuencia del draft	100.000 *drafts*
Mora-Cantallops y Sicilia [21]	Identificación de patrones	MOBA	LoL	Estructura de las redes	Asistencias y asesinatos durante la partida	7.582 partidas
Yu [18]	Predictor de ventaja durante partidas	MOBA	Dota 2	Redes neuronales	Estado de la partida cada 60 segundos	2.802.329 vectores y 71.355 partidas
Mora-Cantallops y Sicilia [28]	Síntesis de trabajos anteriores	MOBA	LoL, Dota 2, Dota, Smite y Heroes of Newerth	Búsqueda y análisis	Trabajos sobre MOBAs	23 trabajos
Kho et al. [22]	Búsqueda de patrones y predictor de victoria	MOBA	LoL	Reglas lógicas, K-satisfacibilidad y redes neuronales	Objetivos tomados y jugadas en una partida	Partidas de la temporada de primavera de 2018 de LCK, LCS y LEC
Lee et al. [19]	Predictor de victoria	MOBA	LoL	Árboles aleatorios	Información durante las partidas	30.108 partidas
Rama et al. [23]	Identificación de patrones	MOBA	LoL	Modelos ocultos de Márkov	Eventos durante la partida	7.620 partidas

Tabla 2.1: Trabajos encontrados en el estudio bibliográfico.

Documento	Tema	Dominio	Juego	Métodos	Datos	Tamaño base de datos
Yang et al. [24]	Identificación de patrones	MOBA	Dota 2	Árboles de decisión	Interacciones entre campeones durante la partida	407 partidas
Costa et al. [29]	Crear composiciones según estrategia	MOBA	LoL	Algoritmo genético	Información sobre los campeones	141 campeones
Myślak y Deja [25]	Sistema de emparejamiento	MOBA	LoL	Regresión logística	Información sobre jugadores	>200.000 partidas
Shores et al. [27]	Identificación de jugadores tóxicos	MOBA	LoL	Métodos estadísticos	Estadísticas de partidas y feedback de los jugadores	2,5 millones de jugadores y 18,25 millones de partidas
Véron et al. [26]	Sistema de emparejamiento	MOBA	LoL	Métodos estadísticos	Información sobre partidas y jugadores	28 millones de sesiones de partidas
Sánchez-Ruiz y Miranda [31]	Predictor de victoria	RTS	StarCraft	LDA, QDA, SVM, KNN y KKNN	Estado de la partida cada 30 segundos	100 partidas de 4 jugadores y 200 de 2 jugadores
Sánchez-Ruiz [32]	Predictor de victoria	RTS	StarCraft	LDA, QDA, SVM, KNN y KKNN	Estado de la partida cada 5 segundos	100 partidas de 2 jugadores
Sánchez-Ruiz [33]	Predictor de victoria	RTS	StarCraft	LDA, QDA, SVM, KNN y KKNN	Estado de la partida cada 6 segundos	200 partidas de 2 jugadores
Volz et al. [34]	Predictor de victoria	RTS	StarCraft II	Árboles de decisión, KNN, árboles aleatorios y redes neuronales	Estados de partida cada 10 segundos	4.012 partidas
Barriga et al. [35]	Inteligencia artificial	RTS	µRTS y Total War: Warhammer	Redes neuronales convolucionales	12 estados aleatorios por partida	2.452 partidas

camente los ids de los campeones obtiene mejores resultados. Un trabajo similar es el que realizan Kinkade y Lim [16]. Crean dos atributos para medir la sinergia entre campeones y cuánto contrarresta un campeón a otro. Comparan el uso de regresión lineal y árboles aleatorios, obteniendo mejores resultados con el primer modelo. Ani et al. [15] también prueba a usar árboles aleatorios además de distintos algoritmos de potenciación, obteniendo un mejor desempeño con los primeros. Como novedad respecto a los datos de entrada, emplean los ids de los campeones, los vetos, los hechizos y la posición. Wang [14] estudia por primera vez usar como modelo las redes neuronales y como atributo la duración de las partidas sin obtener mejoras significativas frente a la regresión logística sin el nuevo atributo. Por último, Semenov et al. [13] estudian el desempeño de distintos algoritmos: Naive Bayes, regresión logística, árboles de decisión potenciados por gradientes y máquinas de factorización, y consiguen los mejores resultados con estos dos últimos. Además, añaden atributos que indican el número de campeones que cumplen cierta función (por ejemplo, apoyar).

Respecto a los recomendadores de campeones, se caracterizan por tener un método de evaluación de las composiciones (predictores de victoria) y un método de búsqueda para encontrar el campeón que aumenta la probabilidad de victoria. Los trabajos que entran en esta división se mencionan a continuación: Conley y Perry [11] prueban dos modelos para evaluar las composiciones: regresión logística y K-NN, consiguiendo los mejores resultados con el primero. Para buscar los posibles campeones, iteran probando con todas las posibilidades. Kalyanaraman [12] mejora este trabajo combinando algoritmo genético con regresión logística. Por otro lado, Hanke y Chaimowicz [8] utilizan reglas de asociación para recomendar campeones. Las reglas de asociación son subconjuntos de campeones que suelen ganar juntos y pares de campeones en el que uno es especialmente fuerte contra el otro (es decir, uno contrarresta al otro). Todos estos trabajos sólo utilizan los ids de los campeones y el equipo ganador.

Sin embargo, ninguno de los trabajos mencionados anteriormente tiene en cuenta las futuras posibles elecciones del rival. Una postura más realista es considerar la selección de campeones como un juego combinatorio donde se tiene en cuenta las acciones futuras de ambos jugadores para elegir el campeón que aumenta tu probabilidad de victoria. En esta postura, da Costa Oliviera et al. [7] emplean minimax (con poda alfa-beta) y regresión lineal para evaluar las posibles composiciones. También, OpenAI [9] utiliza minimax pero con sólo 17 campeones debido a su complejidad computacional. Chen et al. [6] mejora el desempeño de minimax con árboles de búsqueda de Monte Carlo. Prueba tres modelos de clasificación: redes neuronales, regresión logística y árboles de decisión potenciados por gradientes, y obtiene los mejores resultados con redes neuronales y regresión logística. También, Ye et al. [10] emplean árboles de búsqueda de Monte Carlo para su IA acabando con la limitación que tenía OpenAI [9]. Por último, Chen et al. [5] ajustan el problema del *draft* para series de partidas en *Honor of Kings* donde se juega al mejor de N partidas, y los campeones elegidos no pueden ser usados en la siguiente partida. Emplean PUCT (*Polynomial Upper Confidence Trees*), en vez de UCT (*Upper Confidence bounds applied to Trees*). Obtienen mejoras significativas para series de partidas en comparación con [6] mientras que para partidas únicas la mejora es pequeña.

Ninguno de estos trabajos emplea estadísticas propias de cada campeón (como su daño o resistencia). Todos ellos emplean el id de los campeones menos OpenAI que calcula la probabilidad de cada una de las posibles composiciones analizando los primeros frames de sus partidas.

La comparación de estos recomendadores se ilustra en la tabla 2.2.

Se puede observar que el mejor método empleado es el árbol de búsqueda de Monte Carlo con redes neuronales [5, 6, 10]. En este trabajo, se utiliza MCTS UCT basándose sobre todo en [6]. Ye et al. [10] emplea también MCTS UCT pero aporta pocos detalles y los resultados son similares. Respecto a Chen et al. [5], su mayor aportación son las series de partidas dependientes entre ellas y esto no se aplica al LoL. Aunque PUCT aporta cierta mejoría a los resultados de UCT para partidas individuales, se considera que las mayores carencias de estos trabajos no se encuentran en el algoritmo de MCTS y por ello, se usa por simplicidad UCT.

Finalmente, cabe señalar que estos trabajos en específico presentan una limitación al evaluar MCTS. Como se ha explicado en la sección 1.2, para evaluar un árbol Monte Carlo se observa el estado final resultante de realizar las acciones sugeridas. Sin embargo, en este ámbito no se puede conocer al ganador al observar el *draft*. En este caso, hasta que no se juegue la partida no se sabrá qué composición es la ganadora. Por ello, hay que desarrollar un método de evaluación. Los trabajos anteriores utilizan el predictor de victoria empleado en el MCTS para obtener el ganador. Sin embargo, esta solución introduce un prejuicio en la evaluación del modelo que favorecerá a los resultados del Monte Carlo. Se plantea una evaluación alternativa en la sección 4.2 que trata de solucionar esta limitación.

2.2.3. LoL como caso de estudio

Los trabajos sobre el *draft* que usan el LoL como caso de estudio son reducidos [2, 7, 15]. Entre las posibles razones, esto puede deberse a la complejidad del LoL y la carencia de bases de datos públicas del LoL.

Respecto a la complejidad del LoL cabe señalar que cuenta con mayor cantidad de campeones que el resto de MOBAs. El Dota 2 tiene actualmente 121 campeones, sin embargo, en los datasets empleados anteriormente se consideran: 17 [9], 111 campeones [6], 113 [8, 13, 14], y 114 [3]. En el caso de Honor of Kings, existen 105 campeones actualmente y en el trabajo desarrollado [5] contaban con 102 campeones. Mientras tanto, el LoL cuenta actualmente con 155 campeones, de los cuales se tienen en cuenta 154. Esta es la cantidad existente durante la primera mitad de abril, en la versión 11.7. Por tanto, este trabajo considera como mínimo 40 campeones más que el resto de trabajos que no empleen el LoL, con la mayor complejidad que esto supone.

Por otro lado, respecto a bases de datos publicadas de *drafts* del LoL, se encuentran únicamente dos. Shores et al. [27] recogen aproximadamente 18 millones de partidas usando una extensión china

Nombre	Juego	Búsqueda	Evaluación	Datos	Pros	Contras
Chen et al. [5]	Honor of Kings	MCTS PUCT	Redes neuronales	Ids de campeón y victoria	Mejores resultados para series de partidas dependientes de las anteriores. Mejor que minimax.	No considera los vetos
Chen et al. [6]	Dota 2	MCTS UCT	Redes neuronales	Ids de campeón y victoria. También puede usar vetos.	Tiene en cuenta los vetos. Mejor que minimax.	-
Da Costa Oliveira et al. [7]	League of Legends	Minimax con poda alfa-beta	Regresión lineal	Ids de campeón, posiciones y victoria	-	Pocos datos. Minimax no es eficiente. No compara su modelo ni muestra resultados.
Hanke y Chaimowicz [8]	Dota 2	Reglas de asociación	Redes neuronales	Ids de campeón y victoria	Tiene en cuenta el parche actual del juego	No asegura el mejor draft al no tener en cuenta las siguientes elecciones posibles de campeones
OpenAI [9]	Dota 2	Minimax	Redes neuronales	Observaciones del estado de la partida	-	Sólo tiene en cuenta 17 campeones
Ye et al. [10]	Honor of Kings	MCTS UCT	Redes neuronales	Ids de campeón y victoria	Mejor que minimax y evita la simulación de UCT con una red neuronal	No considera los vetos
Conley y Perry [11]	Dota 2	Iterar por todas las posibilidades	Regresión logística y K-NN	Ids de campeón y victoria	-	No es eficiente buscando las posibles elecciones. No tiene en cuenta las siguientes elecciones.
Kalyanaraman [12]	Dota 2	Iterar por todas las posibilidades	Algoritmo genético, regresión logística y combinación de ambas	Ids de campeón y victoria	Mejora la regresión lineal combinándola con algoritmo genético	No es eficiente buscando las posibles elecciones. No tiene en cuenta las siguientes elecciones.

Tabla 2.2: Comparación de los recomendadores.

llamada Duowan. Aunque se menciona que estos datos se pueden encontrar en la página de la extensión, esta no parece relacionada con el LoL. Véron et al. [26] recoge 28 millones de partidas durante algo más de un mes de 2013. Sin embargo, no podemos utilizarla, principalmente porque no cuenta con las posiciones de los campeones elegidos. Tampoco cuenta con vetos y no indica las distintas versiones de las partidas, que habría que inferir por la fecha. Además, se puede considerar que la base de datos es demasiado antigua para desarrollar un trabajo actual.

Por tanto, no se cuentan con bases de datos para el estudio del *draft* pues deberían contar con las posiciones de los campeones y los vetos. El resto de bases de datos que se pueden encontrar por internet tienen varias limitaciones: no son muy extensas, suelen mezclar partidas de distintas versiones del juego y no indican el nivel de habilidad de los jugadores. Todas ellas dificultan la realización de estudios y, en concreto, la mezcla de partidas de distintos parches dificultará encontrar patrones de campeones, pues los campeones más fuertes y las composiciones cambian con la versión.

Pese a todos estos inconvenientes, más los específicos de la obtención de partidas que se tratan en la sección 3.1, se emplea el LoL como caso de estudio por su mayor relevancia en la actualidad, como explicado en la sección 1.1.

En los trabajos anteriores sobre composiciones, Da Costa Oliviera et al. [7] emplean las partidas jugadas en CBLoL (*Campeonato Brasileiro de League of Legends*). Esta base de datos, por ser de una competición, es muy reducida y mezcla partidas de distintos parches. Además de esta carencia, no es un trabajo de referencia al no explicar la configuración de sus datos ni mostrar los resultados de su estudio.

El dataset empleado por Ani et al. [15] también es reducido. Utilizan únicamente 1500 partidas sin especificar nada más que se tratan de partidas profesionales. No entran en detalle en la codificación de datos y la evaluación de sus modelos se realiza con una simple partición de la base de datos en un conjunto de entrenamiento y otro de prueba.

Por último, Hong et al. [2] utilizan 10.314 partidas de la versión 10.5. Entrenan sus modelos utilizando la secuencia de vetos y elecciones del *draft*. Sin embargo, no pueden contar con esta secuencia porque es una limitación actual de la API. Se tratará más en profundidad esta limitación y sus implicaciones en los estudios en la sección 3.1. También, evalúan sus modelos mediante una partición entre datos de entrenamiento y de prueba.

Como se puede observar, las referencias de trabajos que empleen el LoL son muy escasas. Además, todos ellos presentan carencias tanto en el dataset como en su planteamiento. Por esta razón, no se pueden usar como referencia ninguno de los trabajos.

Si se tiene en cuenta otros trabajos que no traten del *draft*, Mora-Cantallops y Sicilia [21] emplean 7.582 partidas, Kho et al. [22] las partidas de la temporada de primavera de 2018, Lee et al. [19], 30.108 y Rama et al. [23], 7.620 partidas. Ninguna de las bases de datos es muy extensa.

Aquí publicamos por primera vez un dataset público y extenso sobre el *draft* en el LoL, que detallaremos en la sección 3.1. El dataset se puede encontrar en Recomendador del LoL.

2.2.4. Limitaciones generales

Nótese que en ningún trabajo se utilizan estadísticas de los campeones para entrenar los modelos. Sólo se encuentran dos trabajos que las hayan intentado añadir. Agarwala y Pearce [17] intentaron conseguir describir el estilo de juego de cada campeón mediante las estadísticas que obtenían en las partidas y Semenov et al. [13] añadieron el número de campeones que tienen cierta función en el equipo (por ejemplo, apoyar).

Respecto a los recomendadores de campeones, una carencia común es que ninguno comprueba que las composiciones creadas sean realistas. En los MOBAs, las composiciones tienen que cumplir ciertos requisitos para que sean óptimas pero el requisito mínimo es que los campeones sean buenos en la posición donde juegan. En la sección 2.1, se desarrollan estos requisitos para el LoL en concreto pero ocurre de forma similar para el resto de MOBAs. Los recomendadores actuales no tienen garantías de que sus campeones cubran todos los roles necesarios. Es el caso de los trabajos de MCTS de referencia [5, 6], que al aplicar al LoL no construyen composiciones realistas. En la sección 4.3, se desarrolla más esta limitación.

En resumen, se plantean distintas representaciones de datos para resolver la carencia general en las configuraciones de datos. Se emplean las estadísticas individuales de los campeones y se estudia cómo funcionan los modelos en comparación con la representación usual de ids. Para ello se emplean redes neuronales y árboles de búsqueda Monte Carlo y se comparan los resultados con el trabajo de referencia [6]. Se trata de resolver carencias específicas de este recomendador proponiendo soluciones para la evaluación de MCTS y se estudia el realismo de las composiciones. Se desarrolla un recomendador de campeones del LoL, con la mayor complejidad que presenta, mejor que los desarrollados hasta ahora en este videojuego. Además, se presenta el primer dataset público del LoL sobre el *draft* con más extensión y atributos que cualquiera de las utilizadas en los trabajos anteriores.

3

Metodología

En esta sección se explica la obtención y los detalles del dataset, las distintas representaciones de datos empleadas, la formulación del *draft* y los modelos empleados para realizar las recomendaciones.

3.1. Dataset

Como se ha explicado en la sección 2.2, no existen actualmente bases de datos publicadas de partidas del LoL. Aquellas que se pueden encontrar en internet tienen varias limitaciones: no son muy extensas, suelen mezclar partidas de distintas versiones del juego y no indican el nivel de habilidad de los jugadores.

En los trabajos sobre el *draft* desarrollados en el LoL, se emplean pocas partidas: 10.314 [2], 1.500 [15] y los realizados durante la liga brasileña de 2017 [7]. Además, los dos últimos mencionados no aseguran que sean partidas de la misma versión.

Para no tener estas carencias, se desarrolla una base de datos propia extensa y consistente. Esta cuenta con un nuevo atributo, la habilidad de los jugadores y todas las partidas son del mismo parche. Este requisito asegura que no hubo cambios importantes en el juego y por tanto, no existen inconsistencias en lo fuertes que son ciertos campeones y composiciones. En caso contrario, el estudio se vería afectado negativamente por estas variaciones.

Para la recopilación de partidas, se emplea la API de Riot Games, la empresa desarrolladora del LoL. La API tiene 11 servidores, uno por región y cuenta con distintas peticiones para la obtención de usuarios y partidas.

El proceso para obtener los datos de una partida no es directo. Primero hay que conocer los ids de las partidas que queremos teniendo en cuenta la habilidad de los jugadores. Para ello, se obtienen los ids de los jugadores, después los ids de sus partidas jugadas en un intervalo de tiempo concreto (para que sean del mismo parche) y por último, se obtienen los detalles de las partidas.

Las partidas obtenidas son de jugadores con la habilidad más alta, de los rangos: Challenger,

Grandmaster, Master y Diamante I, y representan apróximadamente el 0,4 % de los jugadores totales [1].

Por tanto, el proceso de obtención de partidas se desarrolla de la siguiente forma. Por cada región y rango, se realiza una consulta para obtener los ids asociados a los nombres de los jugadores. Después, hay que realizar otra consulta para obtener los ids de las cuentas de los jugadores. Con los ids de las cuentas, se pueden solicitar los ids de las partidas jugadas en las fechas del parche 11.7 y con una última consulta, obtener los datos de la partida. En total, se realizan 4 tipos distintos de consultas.

La API presenta ciertas limitaciones que dificultan la obtención de datos. Para realizar consultas se tiene que contar con una clave. Esta se genera teniendo una cuenta y dura 24 horas. Además, la API tiene dos límites de peticiones: 20 por segundo y 100 por 2 minutos. Como se realizan peticiones sin pausa, la segunda limita las consultas a 0.83 peticiones por segundo (100/120s).

En total, se obtienen 757.837 ids de partidas que se reducen a 433.599 quitando duplicados. Después, se eliminan las partidas que no cumplan el formato de partida clasificatoria (con vetos y sin repeticiones de campeones). Finalmente, se cuenta con 433.356 partidas jugadas del 31 de marzo al 14 de abril de 2021 en todas las regiones del mundo y por los jugadores de mayor habilidad. Todas ellas son del parche 11.7, y por tanto no se produjeron cambios en el juego.

En los datos de una partida se indica la siguiente información relativa al *draft*: los ids de los campeones seleccionados, los ids de los vetados y las posiciones de los campeones. Sin embargo, no se especifican algunos datos importantes. La primera, es que no se indica correctamente la secuencia en la que los campeones han sido seleccionados. Esto imposibilita el uso de modelos que se entrenan mediante secuencias, como LSTM. La segunda, es que las posiciones de los campeones no siempre están bien definidas. Por ejemplo, existen juegos donde los roles asignados dicen que había tres junglas en el equipo. Estas carencias de la API limitan y dificultan realizar estudios usando los datos del LoL.

En el procesamiento de las partidas se obtiene: un vector de ids de campeones por equipo, las posiciones de cada campeón y el ganador de la partida. Como una de las codificaciones necesita que los roles sean correctos, las partidas que no lo cumplen se eliminan. Finalmente, se cuenta con 200.592 partidas que cumplen este requisito, que se guardan en un fichero pickle. También, se obtienen los vetos y la habilidad de los jugadores y, aunque no se utilicen en este trabajo, se pueden encontrar en la base de datos.

En resumen, desarrollamos la primera base de datos pública del *draft* en el LoL. Consta de 200.592 partidas de jugadores de habilidad alta, del parche 11.7 y con los roles correctos. Se puede acceder a ella en Recomendador del LoL.

[1] https://www.leagueofgraphs.com/rankings/rank-distribution

3.2. Representación de datos

Como se ha explicado en la sección 2.2, una de las carencias generales del estado del arte es la limitada variedad de las representaciones de datos pues todas ellas se basan en el uso de los ids de los campeones. Sin embargo, se pueden usar las estadísticas propias de cada campeón para representarlo en vez de usar únicamente su id. A continuación se presenta: la codificación tradicional usada en los trabajos anteriores y las dos nuevas alternativas que planteamos.

Se nombran a los equipos como Azul y Rojo, puesto que el campo del LoL se divide en dos mitades: el lado azul y el rojo.

3.2.1. Representación clásica: Vector de ids

En esta representación, utilizada en [5, 6, 11, 12], la partida se representa como un vector v de 155 valores. Los primeros 154 elementos hacen referencia a los ids de los campeones y el último al equipo ganador.

Formalmente, sea $V \subset \mathbb{Z}^n$ los estados posibles donde $n = 155$ (el número de campeones más uno), se cumple que $v \in V$ si $v_i \in \{-1, +1\}$ cuando $i \in [0, 153]$ y $v_{154} \in \{0, 1\}$.

Para $i \in [0, 153]$, v_i cumple:

$$v_i = \begin{cases} 1, & \text{si el campeón } i \text{ ha sido elegido por el Azul} \\ -1, & \text{si el campeón } i \text{ ha sido elegido por el Rojo} \\ 0, & \text{en otro caso} \end{cases}$$

donde el campeón *i* hace referencia al campeón en la posición *i* del vector de ids de campeones ordenado ascendentemente.

Se aclara el orden de los ids pues estos no siguen la secuencia de los números naturales, existiendo muchos ids mayores que 400.

Cuando $i = 154$ se cumple:

$$v_{154} = \begin{cases} 1, & \text{si equipo Azul ganador} \\ 0, & \text{si equipo Rojo ganador} \end{cases}$$

3.2.2. Estadísticas globales por equipo

En las siguientes dos representaciones se utilizan las estadísticas individuales de cada campeón. En la Tabla 3.1 se indican y describen cada una de ellas, indicando su rango.

Existen 26 atributos asociados a un campeón. De los cuales, 24 son numéricos y 2 categóricos: etiquetas y recurso. Las etiquetas hacen referencia a la clasificación de campeones en 6 grupos: asesinos, luchadores, magos, tiradores, apoyos y tanques, explicada en la sección 2.1. El recurso indica el tipo de medio que se emplea para lanzar habilidades, también explicada en la sección 2.1. Este recurso puede tomar 13 valores distintos.

Las variables categóricas se codifican usando *one-hot*, por tanto, de 26 atributos por campeón se pasa a tener 43 (24 variables numéricas, 6 etiquetas distintas y 13 tipos de recurso).

Se suma por equipo los 43 atributos de sus 5 campeones para obtener los valores acumulados de cada composición.

Por tanto, esta representación consiste en un vector $v \in \mathbb{R}^{87}$ donde:

$$v_i = \begin{cases} \text{atributo equipo Azul}, & \text{si } i \in [0, 42] \\ \text{atributo equipo Rojo}, & \text{si } i \in [43, 85] \end{cases}$$

Cuando $i = 86$ se cumple:

$$v_{86} = \begin{cases} 1, & \text{si equipo Azul ganador} \\ 0, & \text{si equipo Rojo ganador} \end{cases}$$

En el Apéndice C se detallan varios ejemplos de campeones con su representación de estadísticas.

3.2.3. Estadísticas individuales por roles

Similar a la configuración anterior, en este caso se recogen las estadísticas individuales de cada campeón según la posición en el equipo: superior, jungla, medio, inferior y apoyo. Como un campeón tiene 43 atributos, en total son 430 atributos de estadísticas.

Sea el vector de atributos $v \in \mathbb{R}^{431}$ donde:

Nombre	Explicación	Valores
Ataque	Valoración de cuánto ataque proporciona.	[0, 10]
Defensa	Valoración de cuánta defensa proporciona.	[0, 10]
Magia	Valoración de cuánta magia proporciona.	[0, 10]
Dificultad	Valoración de la dificultad de jugarlo.	[0, 10]
Etiquetas	Etiqueta o etiquetas que describen el tipo de campeón. Las posibilidades son: asesino, luchador, tirador, mago, apoyo y tanque.	{Assasin, Fighter, Marksman, Mage, Support, Tank}
Recurso	Indica el tipo de recurso que utiliza para lanzar o potenciar sus habilidades.	{Fury, Ferocity, Heat, Blood Well, Courage, Grit, Flow, Shield, Crimson Rush, Rage, Energy, None, Mana}
Vida base	Cantidad de vida.	[340, 626]
Vida por nivel	Vida adicional por cada nivel.	[65, 115]
Maná base	Recurso que algunos campeones consumen para lanzar habilidades. Puede ser maná o energía.	[0, 10000]
Maná por nivel	Maná adicional por cada nivel.	[0, 87]
Velocidad de movimiento	Velocidad base con la que se mueve.	[315, 355]
Armadura base	Resistencia al daño de ataque. Este se puede infligir por ataques básicos o habilidades.	[17, 47]
Armadura por nivel	Armadura adicional por cada nivel.	[0, 4.3]
Resistencia mágica base	Resistencia al daño mágico. Este daño sólo se puede infligir mediante habilidades.	[25, 39]
Resistencia mágica por nivel	Resistencia adicional por cada nivel.	[0.3, 1.75]
Rango de ataque	Distancia desde la cual se pueden dar ataques básicos.	[125, 650]
Regeneración de vida base	Cantidad de vida que se regenera cada 5 segundos.	[2.5, 10]
Regeneración de vida por nivel	Regeneración de vida adicional por cada nivel.	[0.4, 1.75]
Regeneración de maná base	Cantidad de maná que se regenera cada 5 segundos.	[0, 50]
Regeneración de maná por nivel	Cantidad de maná adicional por cada nivel.	[0, 1]
Golpe crítico base	Probabilidad de que un ataque básico sea un golpe crítico.	0
Golpe crítico por nivel	Golpe crítico adicional por nivel.	0
Daño de ataque básico	Cantidad de daño que inflige un ataque básico.	[40, 72]
Daño de ataque por nivel	Daño de ataque adicional por nivel.	[0, 5]
Velocidad de ataque base	Velocidad con la que realiza ataques básicos.	[0.475, 0.8]
Velocidad de ataque por nivel	Velocidad de ataque adicional por nivel.	[0, 6]

Tabla 3.1: Las estadísticas de un campeón.

$$v_i = \begin{cases} \text{atributo superior Azul}, & \text{si } i \in [0, 42] \\ \text{atributo superior Rojo}, & \text{si } i \in [43, 85] \\ \text{atributo jungla Azul}, & \text{si } i \in [86, 128] \\ \text{atributo jungla Rojo}, & \text{si } i \in [129, 171] \\ \text{atributo medio Azul}, & \text{si } i \in [172, 214] \\ \text{atributo medio Rojo}, & \text{si } i \in [215, 257] \\ \text{atributo inferior Azul}, & \text{si } i \in [258, 300] \\ \text{atributo inferior Rojo}, & \text{si } i \in [301, 343] \\ \text{atributo apoyo Azul}, & \text{si } i \in [344, 386] \\ \text{atributo apoyo Rojo}, & \text{si } i \in [387, 429] \end{cases}$$

Cuando $i = 430$ se cumple:

$$v_{430} = \begin{cases} 1, & \text{si equipo Azul ganador} \\ 0, & \text{si equipo Rojo ganador} \end{cases}$$

En el Apéndice C se detallan varios ejemplos de campeones con su representación de estadísticas.

3.3. Metodología

Para obtener un *draft* con mayores probabilidades de victoria, se construye un modelo que recomiende secuencialmente campeones, que también se emplea en [5, 6]. Existen distintas posibilidades para recomendar como minimax y árboles de búsqueda de Monte Carlo. Se emplea MCTS porque obtiene mejores resultados y en concreto, UCT [6], como explicado en la sección 2.2. Además, cualquier algoritmo de búsqueda necesita una función de evaluación. Las redes neuronales han obtenido mejores resultados, como se explica en la subsección 3.3.3.

A continuación, se expone la formulación del problema del *draft*, el funcionamiento de un árbol de búsqueda de Monte Carlo y la función de evaluación.

Las limitaciones encontradas en la evaluación del Monte Carlo se explican en el capítulo siguiente, en la sección 4.2.

3.3.1. Formulación del problema

El *draft* puede ser definido como un juego combinatorio de dos jugadores, suma cero e información perfecta. El campo del LoL se divide en dos mitades: el lado azul y el rojo. Usando esta distinción se nombra como Azul al jugador que representa al equipo que juega en el lado azul y Rojo, al contrario.

Se puede definir formalmente de la siguiente forma:

- Número de jugadores: $n = 2$.
- El estado del juego: El conjunto de estados posibles es $E \subset \mathbf{Z}^N$. Un estado $e \in E$ es un vector de dimensión N que representa los campeones elegidos por los dos equipos donde N es el número total de campeones posibles. Cada coordenada del vector puede tomar el valor -1, 0 o 1. Los campeones seleccionados por el equipo Azul toman el valor 1, los del equipo Rojo, -1, y los no elegidos, 0. Podemos formular cada coordenada de la siguiente forma:

$$e_i = \begin{cases} 1, & \text{si el campeón } i \text{ ha sido elegido por el Azul} \\ -1, & \text{si el campeón } i \text{ ha sido elegido por el Rojo} \\ 0, & \text{en otro caso} \end{cases}$$

  Cabe señalar que i no representa el id del campeón pues en el LoL no se sigue el orden natural para numerar los ids de los campeones y aún teniendo aproximadamente 150 campeones, hay campeones con id superior a 400. i hace referencia al campeón en la posición i del vector de ids de campeones ordenado ascendentemente.
- El estado inicial: $e = 0^N$, pues todavía no se ha elegido a ningún campeón.
- El estado final: Como una composición la forman cinco campeones, el estado final cuenta con cinco términos con valor 1 y otras cinco con valor -1.
- Las posibles acciones, A: Escoger un campeón entre los que todavía no han sido elegidos. En el primer turno, existen N posibles acciones.
- La función del turno: $t : E \rightarrow (Azul, Rojo)$. Decide a cuál de los equipos le toca elegir siguiendo la secuencia 1-2-2-2-2-1 para alternar entre los dos jugadores.
- La función de recompensa: $R : E \rightarrow \mathbb{R}$ devuelve la probabilidad de victoria del equipo Azul para un estado e final.

3.3.2. Árbol de búsqueda Monte Carlo

El árbol de búsqueda de Monte Carlo [36] es un método para obtener decisiones óptimas secuenciales en problemas que se pueden representar como árboles de búsqueda. A diferencia de otros métodos, Monte Carlo no necesita generar el árbol completo pues emplea muestras aleatorias y sus resultados se mantienen igualmente cercanos a los óptimos. Además, se puede parar la búsqueda

en cualquier momento y devolverá la acción más prometedora que haya encontrado hasta el momento. MCTS ha obtenido numerosos buenos resultados pero en concreto destacan los conseguidos en el juego Go. Por lo expuesto anteriormente, los árboles de Monte Carlo son mejores árboles de búsqueda que otros (como minimax) para el problema del *draft*.

Respecto a su funcionamiento general, en cada iteración se emplea una política para seleccionar los nodos más prioritarios. Esta política intenta balancear la exploración (centrarse en explorar acciones poco conocidas) con la explotación (centrarse en acciones que parecen prometedoras).

Como se ha explicado en la sección 2.2, la versión particular usada de Monte Carlo es *Upper Confidence bound applied to Trees* (UCT). Como su nombre indica, UCT emplea la política UCB1 que se basa en la teoría del bandido multibrazo (*multi-armed bandit*) [36]. La política indica que hay que seleccionar el brazo que maximice

$$UCB1 = \overline{X}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

donde $\overline{X}_j$ es la recompensa del brazo j, n_j es el número de veces que el brazo j ha sido seleccionado y n es el número de veces total que se ha escogido un brazo.

Respecto al funcionamiento del árbol de búsqueda UCT, este se genera mediante los siguientes cuatro pasos que se ejecutan por cada iteración:

- **Selección**: Desde el nodo raíz, se seleccionan nodos hijos sucesivamente hasta alcanzar un nodo que se pueda expandir. La selección se realiza siguiendo el algoritmo UCB1. El nodo j seleccionado maximiza

$$UCT = \overline{X}_j + c\sqrt{\frac{\ln n}{n_j}}$$

  donde $\overline{X}_j$ es la recompensa asociada al nodo j (número de victorias entre número de visitas), n_j es el número de veces que el nodo j ha sido visitado, n es el número de veces total que el nodo padre ha sido seleccionado y la constante c es el término de exploración [6].
- **Expansión**: Se elige aleatoriamente una acción entre las posibles y se crea un nuevo nodo.
- **Simulación**: Se simulan, a partir de este nodo, acciones aleatorias hasta alcanzar un estado final y se evalúa el resultado.
- **Retropropagación**: Se propaga el resultado de la simulación desde el nuevo nodo hasta el nodo raíz. Se actualizan el número de visitas y la media de la recompensa.

3.3.3. Función de evaluación: Red neuronal

La función de evaluación usada anteriormente es, en concreto, un predictor de victoria. Los mejores predictores de los trabajos mencionados en la sección 2.2 se obtienen usando redes neuronales. En dos trabajos [5,6] compararon su rendimiento con regresión logística y árboles de decisión potenciados por gradientes, y obtuvieron peores resultados que con redes neuronales. Por su parte, Wang [14] obtiene resultados parecidos usando regresión logísticas y redes neuronales. Por último, Hong et al. [2] también obtiene los mejores resultados usando redes neuronales frente a árboles aleatorios.

El resto de trabajos no prueban con redes neuronales pero lo hacen con, entre varios modelos, regresión logística [11, 16] o árboles de decisión potenciados por gradientes [13]. Consiguen mejores resultados con estos modelos y como ambos han mostrado peores resultados en comparación con las redes neuronales, se utilizan estas últimas como modelo.

En el caso concreto de los trabajos sobre recomendadores que emplean minimax o MCTS [5–7], se prueban distintos modelos como función de evaluación: regresión logística, redes neuronales y árboles de decisión potenciados por gradientes. Entre ellos, los mejores resultados se obtienen nuevamente con las redes neuronales.

En [6] se emplea un perceptrón multicapa o MLP (*multilayer perceptron*) con una capa oculta con función de activación ReLU y 120 neuronas. La capa de salida cuenta con una neurona y su función de activación es la sigmoide. Emplea la configuración de datos tradicional del vector de ids y obtiene una precisión de 0.65345.

Se usa esta red como base para la creación de las redes neuronales. En la siguiente sección, se explican sus configuraciones, halladas mediante experimentación para optimizar sus resultados.

4

Experimentación

En esta sección se explica toda la experimentación realizada. Primero, se describe el entrenamiento de las redes neuronales asociadas a cada representación de datos. Después, se entrenan árboles de Monte Carlo que tienen como funciones de evaluación las redes neuronales anteriores. Por último, una vez obtenidos tres árboles optimizados, se evalúa su rendimiento.

En todos los pasos, se comparan los resultados obtenidos con los de los trabajos anteriores.

El código se desarrolla en Python y puede encontrarse en Recomendador del LoL.

4.1. Entrenamiento de la función de evaluación

Como se dijo en la sección anterior, como función de evaluación se emplea MLP, que reportó los mejores resultados en [6]. Entrenamos tres redes neuronales, una por cada representación de datos expuestas en la sección 3.2. A continuación, se explica en detalle la optimización de sus hiperparámetros y los resultados obtenidos.

4.1.1. Optimización de hiperparámetros

Para la optimización de hiperparámetros se utiliza búsqueda de cuadrícula junto con validación cruzada de 10 iteraciones. Se realizan tres optimizaciones en total. Las dos primeras optimizan la exactitud y el número máximo de iteraciones de la red neuronal es 500. La tercera optimiza el valor-F y emplea como máximo 1000 iteraciones. Cada una de estas pruebas se desarrollan en tres MLPs, una por cada representación de datos.

Para facilitar la mención de las configuraciones de datos, se llama *vector*, a la que emplea la representación tradicional de ids de campeones, *global*, a la que emplea las estadísticas acumuladas de cada equipo y *roles*, a la que recoge las estadísticas por posición y equipo.

Las dos primeras pruebas de optimización emplean ReLU como función de activación, 500 iteraciones como máximo y emplean la exactitud como método de puntuación.

En la primera prueba, se prueban distintos números de neuronas en una única capa oculta: 90, 100, 110, 120, 150 y 200. En la tabla 4.1, se indican el número de neuronas óptimo para cada configuración y su correspondiente exactitud obtenida en la búsqueda de cuadrícula.

Configuración	Número de neuronas	Exactitud
Vector	100	0.5098
Global	90	0.5115
Roles	110	0.5121

Tabla 4.1: Exactitud y neuronas óptimas al probar distintos números de neuronas en una única capa oculta.

En la segunda prueba, se prueban para un mismo valor de neuronas distintos números de capas ocultas y distintos valores del tamaño del lote. En concreto, se prueban a tener de 1 a 4 capas ocultas con 100 neuronas cada una (100, (100,100), (100,100,100) y (100,100,100,100)) y como tamaños de lote: 70, 100 y 200. Además del valor de la exactitud obtenido en la búsqueda, se realiza una segunda validación cruzada usando una división distinta para obtener una exactitud más realista. Los valores de la exactitud y de los valores óptimos se muestran en la siguiente tabla 4.2.

Configuración	Tamaño del lote	Capas ocultas	Exactitud búsqueda	Exactitud validación
Vector	70	1	0.5109	0.5100
Global	100	1	0.5137	0.5108
Roles	200	1	0.5123	0.5117

Tabla 4.2: Resultados de probar distintos tamaños de lote y número de capas ocultas con 100 neuronas. Se muestra la exactitud de búsqueda y la exactitud obtenida al realizar una validación cruzada con los hiperparámetros obtenidos.

En las pruebas anteriores se descubre que 500 iteraciones son pocas para que converja la red neuronal. Por esta razón, en la tercera prueba se cambia el número de iteraciones máximas a 1000. También, se prueba, además de ReLU, la función de activación sigmoide y se evalúan empleando el valor-F, que aporta más información que la exactitud.

En la tercera prueba se varían, a parte de la función de activación, los valores del número de capas ocultas y del tamaño del lote. En este caso, el número de neuronas de las capas (antes 100) depende de la configuración de datos. Cada configuración de datos usa el mejor valor obtenido en la primera prueba. La representación *vector* emplea 100, la *global*, 90 y la *roles*, 110.

Se halla para cada función de activación los mejores hiperparámetros y se muestran el valor-F obtenido en la búsqueda y el valor-F y la exactitud obtenidos al realizar otra validación cruzada. Los resultados para la función de activación ReLU se muestran en la tabla 4.3 y los de la función sigmoide en 4.4.

Los valores de F1 y exactitud de validación se calculan mediante validación cruzada sobre una

Configuración	Tamaño del lote	Capas ocultas	F1 entrenamiento	F1 validación	Exactitud validación
Vector ReLU	200	1 (100)	0.5117	0.5034	0.5095
Global ReLU	70	1 (90)	0.5083	0.4962	0.5125
Roles ReLU	200	1 (110)	0.5124	0.5042	0.5117

Tabla 4.3: Resultados de probar distintos tamaños de lote y número de capas ocultas. Se emplea el valor-F para la optimización y la función ReLU como función activación.

Configuración	Tamaño del lote	Capas ocultas	F1 entrenamiento	F1 validación	Exactitud validación
Vector S	200	1 (100)	0.5470	0.5293	0.5327
Global S	100	4 (90,90,90,90)	0.5277	0.5290	0.5052
Roles S	100	1 (110)	0.5066	0.5062	0.5059

Tabla 4.4: Resultados de probar distintos tamaños de lote y número de capas ocultas. Se emplea el valor-F para la optimización y la función sigmoide como función activación.

división distinta a la utilizada para optimizar los hiperparámetros.

Los valores hallados con ReLU y la exactitud de sigmoide son bastante estables. El valor-F de validación de sigmoide varía más. El de *vector* es el que más fluctúa, después el de *global* y, por último, el de *roles*. Aunque se han calculado sus valores varias veces, puede existir cierto error en ellos.

Se elige como mejor modelo aquel que tenga mejor valor de F1. En todos los casos, se trata de la función sigmoide. Por tanto, la red neuronal entrenada para la configuración del vector de ids tiene una capa oculta de 100 neuronas y un tamaño de lote 200. La de estadísticas globales cuenta con cuatro capas ocultas de 90 neuronas cada una y un tamaño de lote de 100. Por último, la de estadísticas por roles, cuenta con una capa de 110 neuronas y un tamaño de lote de 100.

Los mejores resultados se obtienen con *vector* y con *global*. Su mejoría respecto a *roles* es clara pero las diferencias no son muy grandes.

De los trabajos desarrollados anteriormente sobre MCTS, sólo Chen et al. [5] calculan el valor-F. Obtienen como valor 0.625. Respecto a la exactitud, obtienen 0.617 y Chen et al. [6] obtienen 0.65345. La diferencia entre estos valores y los obtenidos aquí puede deberse a que ellos cuentan con una base de datos más grande y a que la complejidad de los videojuegos que emplean es menor (*Honor of Kings* y Dota 2). Cabe señalar que la base de datos que aquí se emplea no tiene ninguna composición repetida. Esto indica que no es suficientemente extensa y puede explicar los resultados obtenidos con las redes neuronales. Habría que realizar una hipótesis de contraste para comprobar si las diferencias entre los resultados obtenidos y los de los trabajos son significativas, pero por limitación temporal se

deja como trabajo futuro.

4.2. Entrenamiento de árboles de búsqueda Monte Carlo

Como mencionado anteriormente en las secciones 1.2 y 2.2, una de las limitaciones de los trabajos anteriores sobre MCTS es la evaluación de estos árboles. Para evaluar un árbol Monte Carlo se observa el estado final resultante de realizar las acciones sugeridas. Sin embargo, no se puede conocer al ganador al observar el *draft*. Para obtener el ganador, los trabajos anteriores utilizan el predictor de victoria empleado en el MCTS. Sin embargo, esta solución introduce un prejuicio en la evaluación del modelo que favorecerá a los resultados del Monte Carlo. Por ello, se plantea otra solución.

Se emplean las partidas de las bases de datos para evaluar una composición. La idea es usar las partidas antiguas para evaluar las nuevas composiciones generadas y sacar el porcentaje de victoria. Sin embargo, como la base de datos de este trabajo no es tan extensa, si se compara un *draft* completo con la base de datos es probable que no se obtengan coincidencias. Por esta razón, no se calcula únicamente el porcentaje de victoria si toda la composición es igual.

Cuando se compara una nueva composición con las partidas de la base de datos, se buscan partidas que cumplan:

- Todos los campeones de cada equipo coinciden, es decir, los 5 campeones de cada equipo son iguales.
- 4 de los campeones de cada equipo coinciden, es decir, 8 de ellos son iguales.
- 3 de los campeones de cada equipo coinciden, es decir, 6 son iguales en total.
- 2 de los campeones de cada equipo coinciden, es decir, 4 son iguales en total.

Por tanto, se buscan las partidas que cumplan cuatro requisitos distintos. Por cada conjunto de partidas, se halla el número de victorias, derrotas y el porcentaje de victoria.

Se emplea esta forma de evaluación para optimizar los hiperparámetros de los tres árboles Monte Carlo. Por cada uno de ellos se optimiza el parámetro de exploración c, explicado en la subsección 3.3.2. Como número máximo de iteraciones utilizamos $n = 300$. No se emplean menor número de iteraciones ya que por la complejidad de este problema, se obtendrían peores resultados. Tampoco se emplean mayor número de iteraciones por la restricción de tiempo que existe: una decisión del *draft* tiene que ser tomada en menos de 30 segundos. Si se aumenta el número de iteraciones, el tiempo que tarda el MCTS de la representación de roles supera este límite. Para optimizar el valor de c, se toma $c = 1$ como punto de referencia para comparar su resultado frente a otros valores: $c = \{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 3 \times 2^{-1}\}$ [6]. Los resultados se muestran en las tablas 4.5.

Los porcentajes de victoria se hallan según el número de campeones iguales, como se ha expli-

cado anteriormente. Si una partida tiene coincidencias de 10, 8, 6 y 4 campeones iguales en total, se obtendría un vector de porcentajes de victoria de 4 términos: (0.2, 1, 0.5, 0.6). El primer valor es el porcentaje de victoria de las partidas con 10 campeones iguales y los siguientes con 8 iguales, 6 iguales y 4 iguales. Sin embargo, como es usual que no haya coincidencias con alguna cantidad de campeones, sólo se indica el porcentaje de victoria si hubo coincidencias. Por ejemplo, si sólo hay coincidencias para 6 y 4 campeones iguales en total, se indicará con un vector de dos términos (0.5, 0.6). Si sólo existen coincidencias de 4 campeones iguales, se mostrará un sólo valor, 0.6.

c	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	3×2^{-1}
1	**0.4148**	0.5065	0.5259	0.4820	0.4879	(0, 0.5008)

(a) Porcentajes de *vector*.

c	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	3×2^{-1}
1	0.5081	0.5259	**0.4566**	0.4859	0.4983	0.5165

(b) Porcentajes de *global*.

c	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	3×2^{-1}
1	(1, 0.5409)	0.4955	0.5235	**0.4294**	0.5177	0.5243

(c) Porcentajes de *roles*.

Tabla 4.5: Porcentajes de victoria de c=1 frente a $c = \{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 3 \times 2^{-1}\}$.

El valor de c escogido es el que reduce el porcentaje de victorias de $c = 1$. Por tanto, el valor óptimo de c para *vector* es $c = 2^{-5}$, para *global*, $c = 2^{-3}$ y para *roles*, $c = 2^{-2}$.

4.3. Comparativa

Para evaluar los tres árboles, se realizan dos experimentos. El primero, pone a combatir un mismo árbol consigo mismo. Con esto se quiere ver cuán realistas son las composiciones. En la segunda, los tres árboles compiten entre ellos y se obtienen los porcentajes de victoria de cada combate.

4.3.1. Primer experimento: realismo de las composiciones

Una de las carencias de otros trabajos es que no evalúan si las composiciones son realistas o no. Aquí se realiza este estudio comprobando si las composiciones se ajustan a los roles de una composición (calle superior, jungla, medio, calle inferior y apoyo). Para ello, se generan composiciones enfrentando cada árbol consigo mismo y después se hallan los roles de esas composiciones.

El proceso de hallar los roles se desarrolla de la siguiente forma. Primero, para cada campeón, se obtiene las frecuencias con las que ha sido jugado en cada rol. Después, se elige la mayor frecuencia de todas las halladas entre los 5 campeones. El rol asociado a esa frecuencia, es asignado al campeón.

Tras ello, se vuelve a seleccionar la mayor frecuencia entre los campeones y roles restantes, y se asigna el rol asociado. Se continúa este proceso hasta que todos los roles estén asignados. En el caso de que al seleccionar la frecuencia tenga como valor 0, significa que no se pueden asignar correctamente los roles, pues al menos un campeón tendría un rol asignado que nunca había jugado.

Como estado inicial del árbol, se prueban distintas posibilidades. Se prueba a comenzar con: el primer campeón elegido, 4 campeones elegidos, 6 campeones y 8 campeones. Por cada posibilidad, se crean 10 composiciones distintas. Por tanto, se obtienen 4 vectores de tamaño 10. Los campeones elegidos siguen la distribución de frecuencia de los campeones de la base de datos. Se enfrentan los árboles consigo mismos y se obtienen las composiciones resultantes.

Para evaluar si son realistas las composiciones generadas, se han probado distintos enfoques. El primero, intenta asignar los roles de la composición de cada equipo. Si consigue que ambas composiciones tengan los roles asignados, se evalúa como 1. En caso contrario, como 0. Se calcula este valor por cada posible estado inicial y se halla el promedio de los valores que tuvieran el mismo número de campeones como estado inicial. Los resultados se muestran en la tabla 4.6.

Número campeones	Vector	Global	Roles
1	1	0.92	0.96
4	0.92	0.98	1
6	0.92	1	1
8	1	1	0.98

Tabla 4.6: Promedio de composiciones con roles correctos dependiendo del número de campeones del estado inicial.

Sin embargo, esta postura puede asignar un rol aunque tenga una probabilidad muy pequeña. Por ello, se pone como restricción que todos los roles asignados tienen que tener una probabilidad mayor que 0.1. Se obtienen los resultados de la tabla 4.7.

Número campeones	Vector	Global	Roles
1	0	0	0
4	0	0	0.04
6	0	0	0
8	0	0.02	0.02

Tabla 4.7: Promedio de composiciones con roles correctos dependiendo del número de campeones del estado inicial. Los roles para poder ser asignados tienen que tener una probabilidad mayor a 0.1.

Como se puede observar, el número de composiciones con roles correctos se reduce prácticamente a cero. Para tener más claro qué sucede, se realiza una última prueba donde se obtiene la probabilidad de cada rol asignado. Se obtienen las 5 probabilidades por equipo y se realiza la media. Este valor se halla para todas las composiciones generadas. Por último, como en los otros casos, se calcula el

promedio dependiendo del número de campeones del estado inicial. Los resultados se muestran en la siguiente tabla 4.8.

Número de campeones	p1	p2	p3	p4	p5
1	0.9888	0.9204	0.6439	0.1831	0.0168
2	0.9878	0.9438	0.7175	0.2616	0.0209
3	0.9895	0.9466	0.7374	0.2809	0.0333
4	0.9934	0.9730	0.8793	0.3448	0.0138

(a) Probabilidades de *vector*.

Número de campeones	p1	p2	p3	p4	p5
1	0.9797	0.9326	0.8017	0.3867	0.0527
2	0.9793	0.9302	0.7715	0.3965	0.0520
3	0.9846	0.9412	0.8066	0.4228	0.0430
4	0.9914	0.9678	0.8519	0.4751	0.0509

(b) Probabilidades de *global*.

Número de campeones	p1	p2	p3	p4	p5
1	0.9802	0.9349	0.7543	0.3286	0.0251
2	0.9828	0.9253	0.6939	0.3240	0.0541
3	0.9857	0.9459	0.7844	0.3845	0.0701
4	0.9915	0.9678	0.8393	0.4531	0.0671

(c) Probabilidades de *roles*.

Tabla 4.8: Las probabilidades de que cada campeón sea del rol asignado. p1 es la probabilidad asociada a la primera asignación de campeón y rol, p2 a la segunda, y así sucesivamente.

Las probabilidades de las primeras asignaciones del rol al campeón son muy altas. Disminuye considerablemente al asignar el cuarto campeón y la probabilidad del último es muy pequeña.

Por tanto, se observa que las composiciones no son totalmente realistas aunque se acerquen. Se puede afirmar que se crean composiciones con, al menos, 4 campeones en un rol adecuado. Las representaciones de *global* y de *vector* obtienen mejores resultados que *vector*, pues los porcentajes de las últimas asignaciones son mayores.

4.3.2. Segundo experimento: torneo

Se enfrentan los tres modelos entre sí para obtener sus porcentajes de victoria. En cada comparación, se ejecutan 30 partidas. El primer campeón se elige siguiendo la distribución de frecuencias de los campeones de la base de datos, como hizo [6]. Cada modelo empieza primero eligiendo campeón el mismo número de veces. Los resultados se muestran en la tabla 4.9.

	Vector	Global	Roles
Vector	-	0.5544	0.4893
Global	0.4456	-	0.4950
Roles	0.5107	0.5051	-

Tabla 4.9: Porcentajes de victoria de los modelos de la izquierda frente a los de la derecha.

Tanto *vector* como *roles* ganan a *global*, donde *vector* lo hace de forma significativa. *Roles* supera a *vector* pero lo hace por poco. Es interesante cómo la red neuronal de *roles* obtenía peores resultados que las del resto de representaciones, pero al enfrentar los árboles, sus resultados son un poco mejores que el resto.

4.4. Resultados

Las redes neuronales obtienen mejores resultados en este orden: *vector*, *global* y *roles*. Los valores de las dos primeras son parecidos y ambos se diferencian del de *roles*. *Vector* y *global* obtienen un valor-F de 0.529 aproximadamente y el de *roles* es 0.506.

Con estas redes neuronales, se han optimizado tres árboles y se han enfrentado entre sí. El orden de las configuraciones que obtienen mejores resultados son: *roles*, *vector* y *global*. *Roles* gana a las otras dos configuraciones pero no lo hace de forma significativa (0.5107 y 0.5051). *Vector* gana a *global* de manera más significativa con un porcentaje de victoria de 0.5544.

Respecto al primer experimento, las composiciones no parecen del todo realistas pues uno de los campeones tiene muy poca probabilidad de jugar en ese rol. De todas formas, se puede afirmar que se crean composiciones con, al menos, 4 campeones en un rol adecuado. Se observa que *global* y *roles* lo hacen algo mejor que *vector*. Sobre todo, se puede apreciar en la asignación del cuarto rol con un campeón.

Los resultados serían más determinantes con mayor número de iteraciones. De esta forma, se optimizarían completamente los modelos y se podrían valorar sabiendo que no podrían funcionar mucho mejor.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

El resultado del *draft* en un MOBA es esencial para la victoria de la partida. Por esta razón, se plantea hacer un recomendador de campeones en el LoL, el MOBA más popular. Se desarrolla un estudio bibliográfico general y en específico de los recomendadores, cuyos mejores modelos son los árboles de búsqueda de Monte Carlo junto con redes neuronales. Además, se descubren ciertas carencias en los trabajos anteriores: no existe un dataset público sobre el *draft* del LoL, la evaluación de los árboles de Monte Carlo introduce un prejuicio y no se comprueba si las composiciones son realistas. Sin embargo, la mayor carencia se encuentra en la representación de los datos: sólo se utilizan los ids de los campeones para representarlos. Por esta razón, se desarrollan dos nuevas configuraciones de datos que tienen en cuenta las estadísticas individuales de los campeones. Una recoge las estadísticas acumuladas de cada equipo y otra, las estadísticas de cada rol del equipo. Con ellas, se entrenan redes neuronales y árboles de búsqueda Monte Carlo.

Igualmente, se resuelven el resto de carencias. Se desarrolla una base de datos propia del LoL, que cuenta con los roles de los campeones. La evaluación de los árboles de Monte Carlo se realiza empleando la propia base de datos en vez de su red neuronal. Por último, se comprueba que las composiciones obtenidas con MCTS cumplan los roles de una composición del LoL.

Los mejores resultados de las redes neuronales los obtenemos con la representación tradicional de ids y con la de estadísticas por equipo. Estos valores se diferencian bastante del obtenido con estadísticas por roles. Sin embargo, al aplicarlas a los árboles de búsqueda de Monte Carlo, esta última obtiene mejor porcentaje de victoria que las otras dos, sin ser una diferencia significativa. Por su parte, la representación de ids supera considerablemente a la de estadísticas globales.

Por otro lado, al estudiar si las composiciones son realistas, se obtienen peores resultados con la representación de ids. Tanto la representación de estadísticas global por equipo como la de estadísticas individuales por rol, la supera. Sobre todo, se manifiesta la diferencia en la cuarta asignación de un rol a un campeón. Todas ellas, no desarrollan composiciones realistas, pues el campeón al que se le asigna el último el rol tiene muy poca probabilidad de jugar en él. Se puede afirmar que se crean

composiciones con, al menos, 4 campeones en un rol adecuado.

No se puede llegar a la conclusión de descartar una de las dos posturas de representaciones: usar los ids o usar estadísticas. Sin embargo, las nuevas representaciones obtienen resultados parecidos y deberían considerarse para trabajos futuros como se considera la representación clásica. Un estudio que emplee mayor número de partidas para entrenar las redes neuronales y mayor número de iteraciones, podría esclarecer esta cuestión.

Los motivos por los que algunos resultados no son concluyentes pueden ser debido a que la complejidad del problema es muy grande y la base de datos no es suficientemente extensa. Los trabajos que desarrollan un estudio similar, usando MCTS, emplean bases de datos con millones de partidas. Aunque una base de este tamaño está fuera del alcance de este trabajo, las aportaciones presentadas aquí pueden usarse de referencia para el desarrollo de trabajos futuros. En concreto, para: el *draft* en el LoL, emplear nuevas configuraciones de datos y evaluar de nuevas formas los árboles de Monte Carlo y las composiciones.

5.2. Trabajo futuro

En el trabajo futuro se puede desarrollar este estudio con ciertas modificaciones que permitan obtener resultados más precisos dentro de la complejidad que supone el *draft*. Cabe señalar que la base de datos utilizada no tiene ninguna composición repetida. Esto indica que no es suficientemente extensa para cubrir las posibilidades existentes en el LoL. Por esto, se considera que emplear una base de datos más extensa permitiría obtener mejores resultados. En concreto, podría mejorar los resultados de las redes neuronales hasta alcanzar los valores de los otros trabajos.

Así mismo, se puede mejorar el tiempo que tarda el MCTS en recomendar un campeón modificando el código o ejecutándolo en un dispositivo de mejores características. De esta forma, tendría sentido aumentar el número máximo de iteraciones del árbol y se obtendrían mejores resultados. Además, la comparación entre árboles sería más realista al estar completamente optimizados.

Bibliografía

[1] A. Summerville, M. Cook, and B. Steenhuisen, "Draft-analysis of the ancients: Predicting draft picks in dota 2 using machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 12, 2016.

[2] S.-J. Hong, S.-K. Lee, and S.-I. Yang, "Champion recommendation system of league of legends," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1252–1254, 2020.

[3] C. Yu, W.-n. Zhu, and Y.-m. Sun, "E-sports ban/pick prediction based on bi-lstm meta learning network," in *Artificial Intelligence and Security*, pp. 97–105, Springer International Publishing, 2019.

[4] D. Gourdeau and L. Archambault, "Discriminative neural network for hero selection in professional heroes of the storm and dota 2," *IEEE Transactions on Games*, 2020.

[5] S. Chen, M. Zhu, D. Ye, W. Zhang, Q. Fu, and W. Yang, "Which heroes to pick? learning to draft in moba games with neural networks and tree search," 2020.

[6] Z. Chen, T.-H. D. Nguyen, Y. Xu, C. Amato, S. Cooper, Y. Sun, and M. S. El-Nasr, "The art of drafting," in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 200–208, 2018.

[7] V. da Costa Oliveira, B. J. Placides, M. de Freitas Oliveira Baffa, and A. Fernandes da Veiga Machado, "A hybrid approach to build automatic team composition in league of legends," in *Proceedings of XVI SBGames*, 2017.

[8] L. Hanke and L. Chaimowicz, "A recommender system for hero line-ups in moba games," 2017.

[9] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019.

[10] D. Ye, G. Chen, W. Zhang, S. Chen, B. Yuan, B. Liu, J. Chen, Z. Liu, F. Qiu, H. Yu, Y. Yin, B. Shi, L. Wang, T. Shi, Q. Fu, W. Yang, L. Huang, and W. Liu, "Towards playing full moba games with deep reinforcement learning," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 621–632, Curran Associates, Inc., 2020.

[11] K. Conley and D. Perry, "How does he saw me? a recommendation engine for picking heroes in dota 2: Semantic scholar," tech. rep., Stanford University, 2013.

[12] K. Kalyanaraman, "To win or not to win ? a prediction model to determine the outcome of a dota 2 match," tech. rep., University of California, San Diego, 2014.

[13] A. Semenov, P. Romov, S. Korolev, D. Yashkov, and K. Neklyudov, "Performance of machine learning algorithms in predicting game outcome from drafts in dota 2," in *Communications in Computer*

and Information Science Analysis of Images, Social Networks and Texts, p. 26–37, 2016.

[14] W. Wang, "Predicting multiplayer online battle arena (moba) game outcome based on hero draft data," tech. rep., National College of Ireland, 2016.

[15] R. Ani, V. Harikumar, A. K. Devan, and O. S. Deepa, "Victory prediction in league of legends using feature selection and ensemble methods," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pp. 74–77, 2019.

[16] N. Kinkade and K. y. K. Lim, "Dota 2 win prediction," tech. rep., University of California, San Diego, 2015.

[17] A. Agarwala and M. Pearce, "Learning dota 2 team compositions," tech. rep., Stanford University, 2014.

[18] L. Yu, D. Zhang, X. Chen, and X. Xie, "Moba-slice: A time slice based evaluation framework of relative advantage between teams in moba games," in *Computer Games* (T. Cazenave, A. Saffidine, and N. Sturtevant, eds.), (Cham), pp. 23–40, Springer International Publishing, 2019.

[19] S.-K. Lee, S.-J. Hong, and S.-I. Yang, "Predicting game outcome in multiplayer online battle arena games," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1261–1263, 2020.

[20] N. Pobiedina, J. Neidhardt, M. d. C. Calatrava Moreno, L. Grad-Gyenge, and H. Werthner, "On successful team formation: Statistical analysis of a multiplayer online game," in *2013 IEEE 15th Conference on Business Informatics*, pp. 55–62, 2013.

[21] M. Mora-Cantallops and M. Ángel Sicilia, "Team efficiency and network structure: The case of professional league of legends," *Social Networks*, vol. 58, pp. 105–115, 2019.

[22] L. C. Kho, M. S. M. Kasihmuddin, M. Mansor, S. Sathasivam, *et al.*, "Logic mining in league of legends.," *Pertanika Journal of Science & Technology*, vol. 28, no. 1, pp. 211–225, 2020.

[23] A. M. Rama, V. Rodriguez-Fernandez, and D. Camacho, "Finding behavioural patterns among league of legends players through hidden markov models," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pp. 419–430, Springer, 2020.

[24] P. Yang, B. Harrison, and D. L. Roberts, "Identifying patterns in combat that are predictive of success in moba games," in *Proceedings of the Foundations of Digital Games 2014 Conference*, 2014.

[25] M. Myślak and D. Deja, "Developing game-structure sensitive matchmaking system for massive-multiplayer online games," in *Social Informatics* (L. M. Aiello and D. McFarland, eds.), (Cham), pp. 200–208, Springer International Publishing, 2015.

[26] M. Véron, O. Marin, and S. Monnet, "Matchmaking in multi-player on-line games: Studying user traces to improve the user experience," 2014.

[27] K. Shores, Y. He, K. L. Swanenburg, R. Kraut, and J. Riedl, "The identification of deviance and its impact on retention in a multiplayer game," in *In Proc. CSCW*, 2014.

[28] M. Mora-Cantallops and M. Ángel Sicilia, "Moba games: A literature review," *Entertainment Computing*, vol. 26, pp. 128–138, 2018.

[29] L. M. Costa, A. C. C. Souza, and F. C. M. Souza, "An approach for team composition in league

of legends using genetic algorithm," in *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 52–61, 2019.

[30] H. N. Ward, D. J. Brooks, D. Troha, B. Mills, and A. S. Khakhalin, "AI solutions for drafting in magic: the gathering," *ArXiv*, vol. abs/2009.00655, 2020.

[31] A. A. Sánchez-Ruiz and M. Miranda, "A machine learning approach to predict the winner in starcraft based on influence maps," *Entertain. Comput.*, vol. 19, pp. 29–41, 2017.

[32] A. A. Sánchez-Ruiz, "Predicting the winner in two player starcraft games," in *Proceedings 2st Congreso de la Sociedad Española para las Ciencias del Videojuego, Barcelona, Spain, June 24, 2015* (D. Camacho, M. A. Gómez-Martín, and P. A. González-Calero, eds.), vol. 1394 of *CEUR Workshop Proceedings*, pp. 24–35, CEUR-WS.org, 2015.

[33] A. A. Sánchez-Ruiz, "Predicting the outcome of small battles in starcraft," in *Workshop Proceedings from The Twenty-Third International Conference on Case-Based Reasoning (ICCBR 2015), Frankfurt, Germany, September 28-30, 2015* (J. Kendall-Morwick, ed.), vol. 1520 of *CEUR Workshop Proceedings*, pp. 33–42, CEUR-WS.org, 2015.

[34] V. Volz, M. Preuss, and M. K. Bonde, "Towards embodied starcraft ii winner prediction," in *Computer Games* (T. Cazenave, A. Saffidine, and N. Sturtevant, eds.), (Cham), pp. 3–22, Springer International Publishing, 2019.

[35] N. A. Barriga, M. Stanescu, F. Besoain, and M. Buro, "Improving rts game ai by supervised policy learning, tactical search, and deep reinforcement learning," *IEEE Computational Intelligence Magazine*, vol. 14, no. 3, pp. 8–18, 2019.

[36] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

Apéndices

A

Partida de League of Legends

Después de la selección de campeones, se desarrolla la segunda fase de la partida. Los equipos aparecen en el mapa y comienza la lucha por ver quién es el primero en destruir la base enemiga.

Para poder conseguirlo, existen objetivos secundarios necesarios para la victoria. Se pueden dividir en: súbditos, estructuras defensivas y monstruos de la jungla.

Los súbditos, o *minions* son pequeños soldados que se generan cada 30 segundos en ambas bases, recorren las tres líneas o calles del mapa y atacan a los campeones, estructuras o súbditos rivales que se encuentren. Los súbditos de tu equipo te facilitan la destrucción de las estructuras rivales. Los súbditos de tu rival son tu principal fuente de oro y experiencia. Obtienes oro si les proporcionas el último golpe necesario para que mueran y recibes experiencia sólo con estar cerca de ellos cuando mueren.

Entre los monstruos de la jungla, se encuentran pequeños monstruos repartidos en campamentos que proporcionan oro y experiencia al jungla (el jugador que no se encuentra en ninguna de las calles). También, existen tres objetivos que proporcionan ventajas al equipo: el Heraldo, el Dragón y el Barón Nashor. El Heraldo facilita la destrucción de estructuras defensivas. Los dragones proporcionan mejoras al equipo dependiendo del elemento al que pertenezcan entre cuatro posibilidades. Por ejemplo, el Dragón de Montaña proporciona armadura y resistencia mágica. El Barón Nashor proporciona una mejora temporal a todo el equipo que otorga más daño y mejora enormemente a los súbditos, facilitando la destrucción de estructuras defensivas.

Respecto a las estructuras defensivas encontramos por línea y equipo, tres torres y un inhibidor. La destrucción de un inhibidor enemigo proporciona mayor poder a tus súbditos de esa calle. En la base de cada equipo, se encuentra el nexo protegido por dos torres. La destrucción del nexo rival supone la victoria del equipo. Podemos observar las estructuras en la imagen A.1.

La principal ventaja en la partida la genera la diferencia de oro respecto al enemigo. El oro se emplea para comprar hasta 6 objetos por jugador. Si tu equipo tiene más oro, tiene más objetos y por tanto, mejores estadísticas. Tu equipo puede tener entre otros: más daño, más resistencia y mayor velocidad de ataque, aumentado así su probabilidad de éxito en futuras peleas o obtención de objetivos.

Figura A.1: Mapa donde se destacan las estructuras que hay que destruir para acabar la partida: los nexos. También, si nos centramos en la calle central del equipo azul, se puede observar que desde su nexo al centro del mapa existen: dos torres protegiendo el nexo, un inhibidor protegido por una torre y dos torres más. *Imagen extraída de Página oficial del League of Legends.*

B

Estudio bibliográfico

Al principio, se emplea Google Scholar [1] para la búsqueda de trabajos. Las consultas que se prueban son: “League of Legends”, “League of Legends draft” y “MOBA draft”.

A partir de los trabajos encontrados por las consultas, se desarrolla un proceso iterativo. Por cada trabajo encontrado relacionado con el *draft*, se revisan sus citas para añadir nuevos trabajos relevantes a la bibliografía. Se vuelve a buscar en las citas de los nuevos y así sucesivamente, hasta que no se encuentren trabajos nuevos.

Además, de esta búsqueda recursiva, por recomendación de la tutora se revisan los trabajos sobre RTS de Santiago Ontañón, Antonio A. Sánchez-Ruiz y Maximiliano Miranda.

Tras este proceso, se obtienen 35 trabajos relativos a videojuegos online con estrategia, de los cuales 29 son relativos a los MOBAs. Los trabajos finales se reducen a 17, aquellos que tratan la composición de campeones.

[1] https://scholar.google.com/

C

Representación de campeones

Aquí se desarrolla más detalladamente cómo se realiza la representación de datos de estadísticas globales y estadísticas individuales por roles. Para ello, se presenta la representación que se haría de un equipo si este estuviera formado por dos campeones.

Los campeones a utilizar son: Ashe, en la posición inferior, y Lux, en la posición de apoyo. Las estadísticas de Ashe y Lux se indican en la siguiente tabla C.1. Se muestran las originales y las transformadas al pasar a *one-hot encoding* las etiquetas y los recursos.

Para la representación de estadísticas globales se suman todas las estadísticas de los campeones de un equipo. En este caso, se suman las de Ashe y Lux, aunque en la realidad se sumarían las de 5 campeones. El nombre de los atributos se consiguen añadiendo el color del equipo a los nombres originales. El resultado final se muestra en la tabla C.2. Las estadísticas del equipo rojo tendrían un representación similar a esta.

Para la representación por roles, las estadísticas se asignan a una posición. En este caso, Ashe es del carril inferior que se nombra como DUO_CARRY y Lux es el apoyo, representado como DUO_SUPPORT. Además, se presupone que son del equipo Azul. La representación se muestra en la tabla C.3. Los roles que faltan para completar la representación de un equipo son: carril superior (TOP), jungla (JUNGLE) y carril medio (MID).

	Ashe	Lux
Ataque	7	2
Defensa	3	4
Magia	2	9
Dificultad	4	5
Etiquetas	[Marksman, Support]	[Mage, Support]
Recurso	Mana	Mana
Vida base	570	490
Vida por nivel	87	85
Maná base	280	480
Maná por nivel	32	23.5
Velocidad de movimiento	325	330
Armadura base	26	19
Armadura por nivel	3.4	4
Resistencia mágica base	30	30
Resistencia mágica por nivel	0.5	0.5
Rango de ataque	600	550
Regeneración de vida base	3.5	5.5
Regeneración de vida por nivel	0.55	0.55
Regeneración de maná base	6.97	8
Regeneración de maná por nivel	0.4	0.8
Golpe crítico base	0	0
Golpe crítico por nivel	0	0
Daño de ataque básico	59	54
Daño de ataque por nivel	2.96	3.3
Velocidad de ataque base	3.33	1
Velocidad de ataque por nivel	658	669

(a) Las estadísticas originales.

	Ashe	Lux
Assassin	0	0
Fighter	0	0
Marksman	1	0
Mage	0	1
Support	1	1
Tank	0	0
Fury	0	0
Ferocity	0	0
Heat	0	0
Blood Well	0	0
Courage	0	0
Grit	0	0
Flow	0	0
Shield	0	0
Crimson Rush	0	0
Rage	0	0
Energy	0	0
None	0	0
Mana	1	1
attack	7	2
defense	3	4
magic	2	9
difficulty	4	5
hp	570	490
hpperlevel	87	85
mp	280	480
mpperlevel	32	23.5
movespeed	325	330
armor	26	19
armorperlevel	3.4	4
spellblock	30	30
spellblockperlevel	0.5	0.5
attackrange	600	550
hpregen	3.5	5.5
hpregenperlevel	0.55	0.55
mpregen	6.97	8
mpregenperlevel	0.4	0.8
crit	0	0
critperlevel	0	0
attackdamage	59	54
attackdamageperlevel	2.96	3.3
attackspeedperlevel	3.33	1
attackspeed	658	669

(b) Las estadísticas procesadas.

Tabla C.1: Estadísticas de Ashe y Lux.

Assassin_blue	0
Fighter_blue	0
Marksman_blue	1
Mage_blue	1
Support_blue	2
Tank_blue	0
Fury_blue	0
Ferocity_blue	0
Heat_blue	0
Blood Well_blue	0
Courage_blue	0
Grit_blue	0
Flow_blue	0
Shield_blue	0
Crimson Rush_blue	0
Rage_blue	0
Energy_blue	0
None_blue	0
Mana_blue	2
attack_blue	9
defense_blue	7
magic_blue	11
difficulty_blue	9
hp_blue	1060
hpperlevel_blue	172
mp_blue	760
mpperlevel_blue	55.5
movespeed_blue	655
armor_blue	45
armorperlevel_blue	7.4
spellblock_blue	60
spellblockperlevel_blue	1
attackrange_blue	1150
hpregen_blue	9
hpregenperlevel_blue	1.1
mpregen_blue	14.97
mpregenperlevel_blue	1.2
crit_blue	0
critperlevel_blue	0
attackdamage_blue	113
attackdamageperlevel_blue	6.26
attackspeedperlevel_blue	4.33
attackspeed_blue	1327

Tabla C.2: Representación de estadísticas globales para equipo Azul formado por Ashe y Lux.

Assassin_blue_DUO_CARRY	0
Fighter_blue_DUO_CARRY	0
Marksman_blue_DUO_CARRY	1
Mage_blue_DUO_CARRY	0
Support_blue_DUO_CARRY	1
Tank_blue_DUO_CARRY	0
Fury_blue_DUO_CARRY	0
Ferocity_blue_DUO_CARRY	0
Heat_blue_DUO_CARRY	0
Blood Well_blue_DUO_CARRY	0
Courage_blue_DUO_CARRY	0
Grit_blue_DUO_CARRY	0
Flow_blue_DUO_CARRY	0
Shield_blue_DUO_CARRY	0
Crimson Rush_blue_DUO_CARRY	0
Rage_blue_DUO_CARRY	0
Energy_blue_DUO_CARRY	0
None_blue_DUO_CARRY	0
Mana_blue_DUO_CARRY	1
attack_blue_DUO_CARRY	7
defense_blue_DUO_CARRY	3
magic_blue_DUO_CARRY	2
difficulty_blue_DUO_CARRY	4
hp_blue_DUO_CARRY	570
hpperlevel_blue_DUO_CARRY	87
mp_blue_DUO_CARRY	280
mpperlevel_blue_DUO_CARRY	32
movespeed_blue_DUO_CARRY	325
armor_blue_DUO_CARRY	26
armorperlevel_blue_DUO_CARRY	3.4
spellblock_blue_DUO_CARRY	30
spellblockperlevel_blue_DUO_CARRY	0.5
attackrange_blue_DUO_CARRY	600
hpregen_blue_DUO_CARRY	3.5
hpregenperlevel_blue_DUO_CARRY	0.55
mpregen_blue_DUO_CARRY	6.97
mpregenperlevel_blue_DUO_CARRY	0.4
crit_blue_DUO_CARRY	0
critperlevel_blue_DUO_CARRY	0
attackdamage_blue_DUO_CARRY	59
attackdamageperlevel_blue_DUO_CARRY	2.96
attackspeedperlevel_blue_DUO_CARRY	3.33
attackspeed_blue_DUO_CARRY	658

(a) Las estadísticas del campeón de la calle inferior, Ashe.

Assassin_blue_DUO_SUPPORT	0
Fighter_blue_DUO_SUPPORT	0
Marksman_blue_DUO_SUPPORT	0
Mage_blue_DUO_SUPPORT	1
Support_blue_DUO_SUPPORT	1
Tank_blue_DUO_SUPPORT	0
Fury_blue_DUO_SUPPORT	0
Ferocity_blue_DUO_SUPPORT	0
Heat_blue_DUO_SUPPORT	0
Blood Well_blue_DUO_SUPPORT	0
Courage_blue_DUO_SUPPORT	0
Grit_blue_DUO_SUPPORT	0
Flow_blue_DUO_SUPPORT	0
Shield_blue_DUO_SUPPORT	0
Crimson Rush_blue_DUO_SUPPORT	0
Rage_blue_DUO_SUPPORT	0
Energy_blue_DUO_SUPPORT	0
None_blue_DUO_SUPPORT	0
Mana_blue_DUO_SUPPORT	1
attack_blue_DUO_SUPPORT	2
defense_blue_DUO_SUPPORT	4
magic_blue_DUO_SUPPORT	9
difficulty_blue_DUO_SUPPORT	5
hp_blue_DUO_SUPPORT	490
hpperlevel_blue_DUO_SUPPORT	85
mp_blue_DUO_SUPPORT	480
mpperlevel_blue_DUO_SUPPORT	23.5
movespeed_blue_DUO_SUPPORT	330
armor_blue_DUO_SUPPORT	19
armorperlevel_blue_DUO_SUPPORT	4
spellblock_blue_DUO_SUPPORT	30
spellblockperlevel_blue_DUO_SUPPORT	0.5
attackrange_blue_DUO_SUPPORT	550
hpregen_blue_DUO_SUPPORT	5.5
hpregenperlevel_blue_DUO_SUPPORT	0.55
mpregen_blue_DUO_SUPPORT	8
mpregenperlevel_blue_DUO_SUPPORT	0.8
crit_blue_DUO_SUPPORT	0
critperlevel_blue_DUO_SUPPORT	0
attackdamage_blue_DUO_SUPPORT	54
attackdamageperlevel_blue_DUO_SUPPORT	3.3
attackspeedperlevel_blue_DUO_SUPPORT	1
attackspeed_blue_DUO_SUPPORT	669

(b) Las estadísticas del campeón de apoyo, Lux.

Tabla C.3: Representación de estadísticas por roles del equipo Azul.